

IBM Integration Bus

WESB to IIB Conversion Feature

using the Stock Quote application

September, 2013

Hands-on lab built at product
code level Version 9.0.0.0

1. INTRODUCTION	3
1.1 STOCKQUOTE APPLICATION OVERVIEW	3
1.2 LAB PREPARATION.....	5
2. IMPORT THE STOCK APPLICATION INTO THE INTEGRATION TOOLKIT	6
3. UPDATE THE GENERATED STOCKQUOTE SERVICE	24
4. TEST THE IIB_STOCKQUOTE SERVICE	48
5. APPENDIX I	50

1. Introduction

IBM Integration Bus version 9 has introduced a conversion utility to assist with the conversion of WebSphere Enterprise Service Bus applications to Integration Bus applications.

The objective of this lab is intended to show the basic framework for these conversion capabilities for WESB applications by using the Stock Quote sample application, packaged in WebSphere Integration Developer as a WESB Sample.

This lab has three primary tasks:

1. Import the Stock Quote sample application into the Integration Bus Toolkit.
2. Run the conversion editor.
3. Perform a number of remaining conversion tasks to make Stock Quote a working service in Integration Bus version 9.

1.1 StockQuote Application overview

The Stock Quote sample addresses the business need of a financial services company that provides an interactive web-based stock market service to its customers.

The company wants to differentiate itself from its competition by offering tiered levels of service. Using the following requirements:

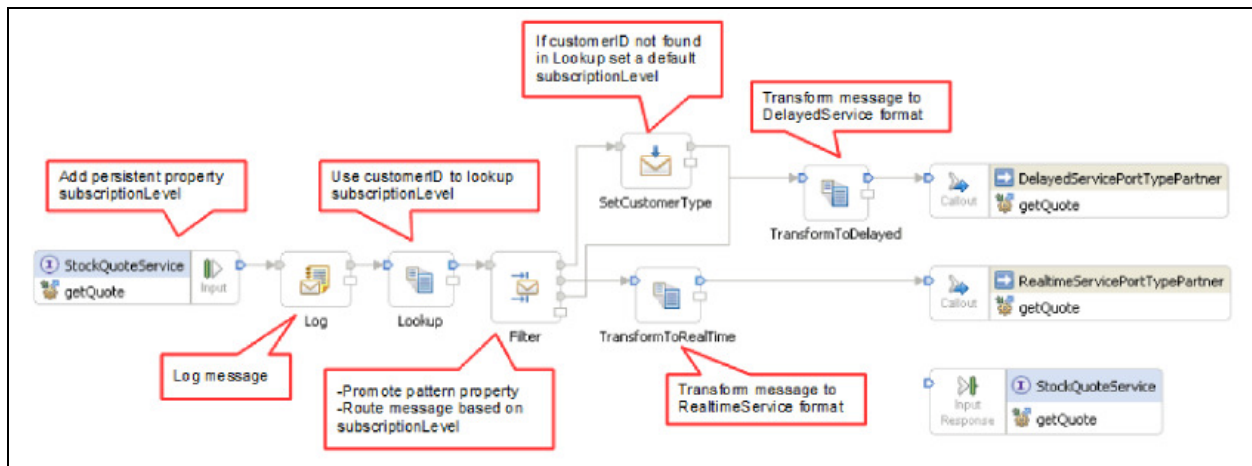
- Offer the delayed and real-time stock quote services as a single service, which dynamically determines which external service to invoke based on the customer's subscription level.
- If the real-time service is unavailable, route requests to the delayed service without affecting the running application.
- Log all requests to the service to satisfy audit requirements

The mediation service that runs on the WebSphere ESB or IBM Process Server is contained in a single mediation module called StockQuote.

The StockQuote mediation module consists of the following elements:

- StockQuoteService has a WSDL interface, called StockQuoteService, and uses SOAP/HTTP.
- StockQuote_MediationFlow contains the mediation flow.
- RealtimeService and DelayedService has a web service binding and an interface that matches the real-time (premium) and delayed (regular) service.

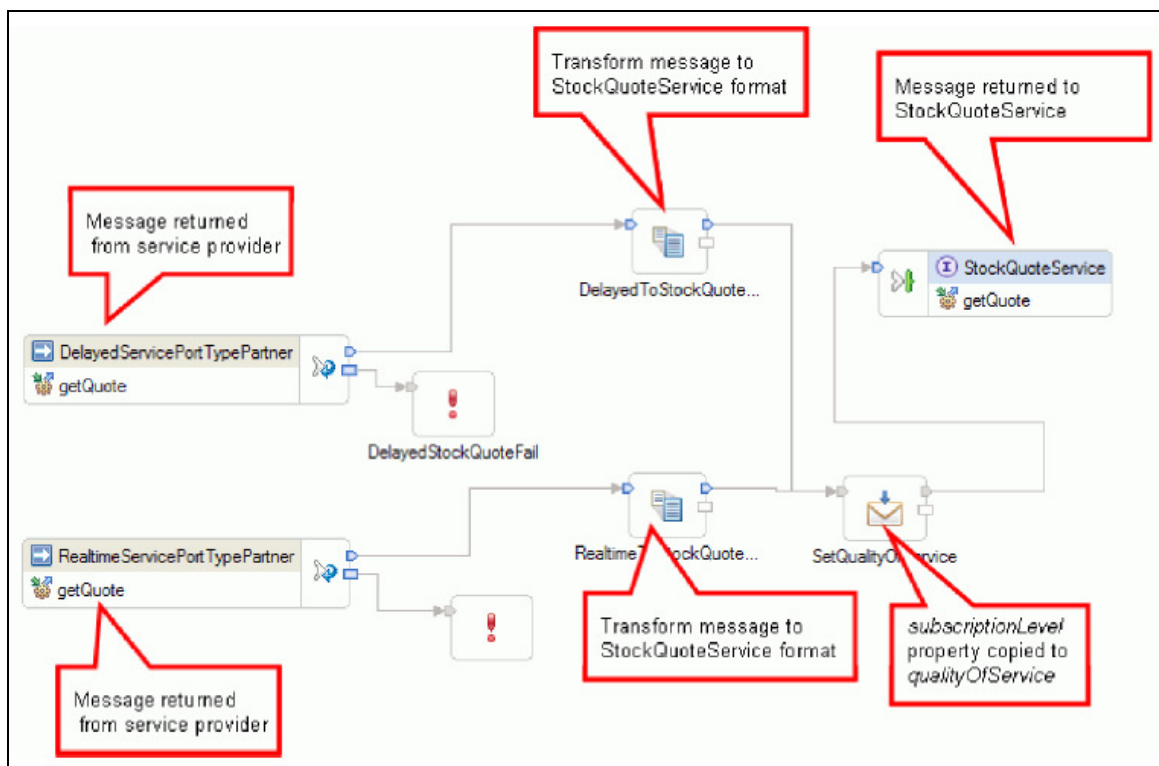
The diagram below shows the request flow that defines the mediation logic applied to the message as it flows through the StockQuote_MediationFlow component to the target service providers.



The diagram below shows the response flow that defines the mediation logic applied to the returning message as it flows through the StockQuote_MediationFlow component from the target service provider to the client. A Message Element Setter is used to copy the value of subscriptionLevel from the correlation context to the property qualityOfService in the message.

The qualityOfService text indicates "Premium" to a response that is returned from the real-time service, and "Standard" to a response that is returned from the delayed service.

The qualityOfService text is displayed in the client to indicate the service provider that was used.



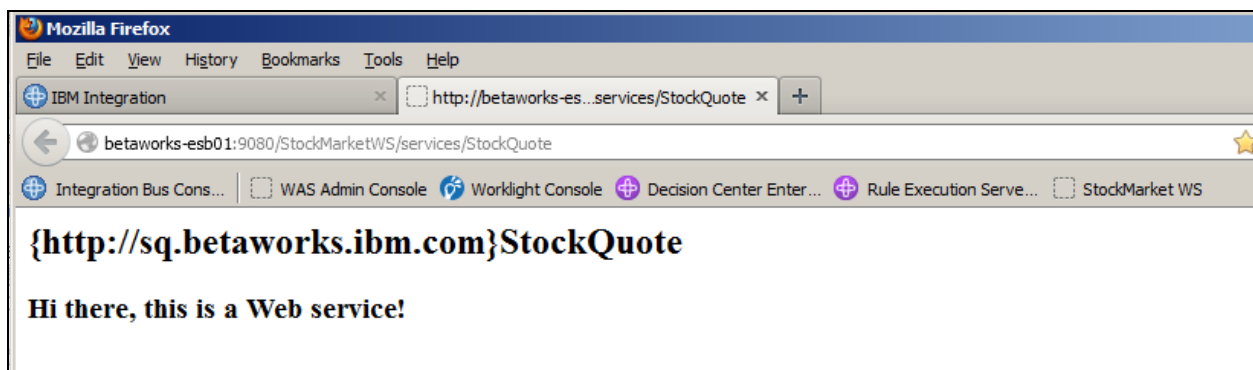
1.2 Lab preparation

This lab will use a dummy application running on WebSphere Application Server. Make sure that ODM WAS is running.

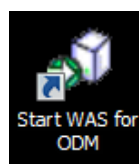
Open a Firefox browser and click the bookmark for “**StockMarket WS**”:



If it is running, you will see this response:



If it is not running, start using the icon on the desktop:

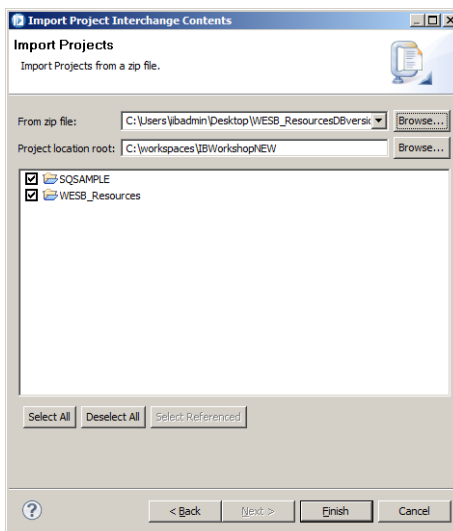


2. Import the Stock Application into the Integration Toolkit

1. In the Integration Toolkit, import the WESB Resource Library. This library contains a database definition file and message test files that are necessary for the StockQuote application.

Right-click on a blank part of the Application Development view, and select Import...> Other > **Project Interchange**.

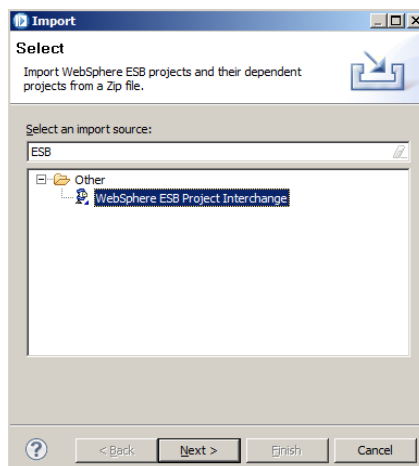
Import the PI file C:\student\WESB_Conversion\resources\WESB_Resources.zip.



This will import a library “WESB_Resources”, and a referenced project SQSAMPLE. Click Finish.

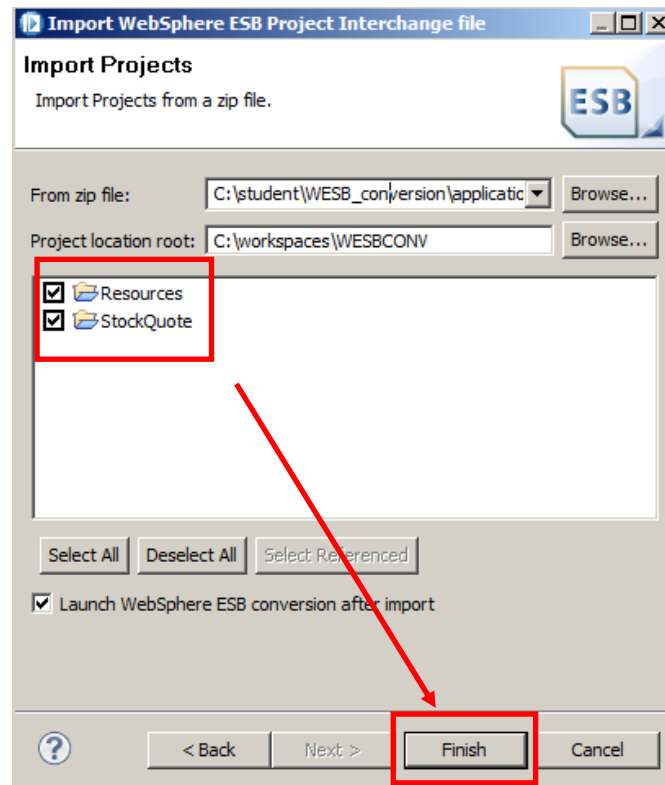
2. Now import the Stock Quote sample application.

Open the Import wizard again. This time, select the “WebSphere ESB project Interchange” item. (You can locate this quickly by typing ESB into the “Select an import source” box. Or go to File > Import > Other > **WebSphere ESB project Interchange**:



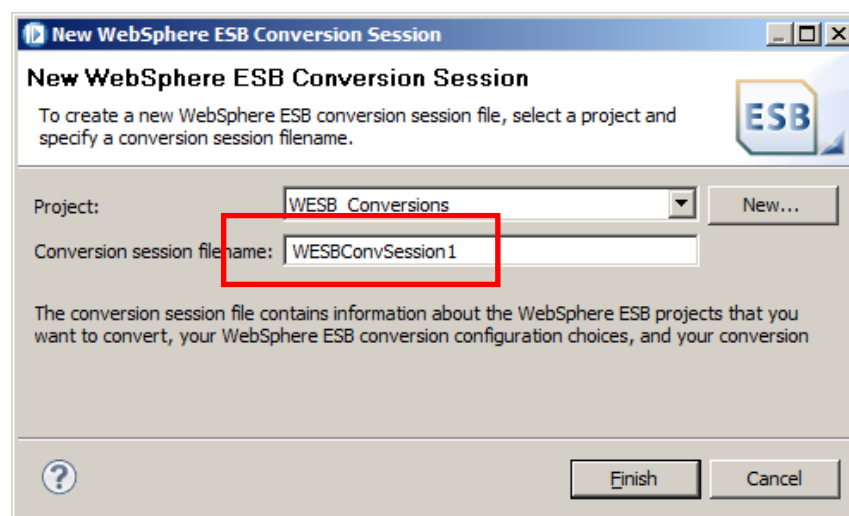
Click Next.

- Use the Browse button to set the import file to
“C:\student\WESB_conversion\applications\PI_StockQuote_IID8.zip”



Select both the “Resources” and “StockQuote” items. **Make sure “Launch WebSphere ESB conversion after import” box is selected.** Click Finish.

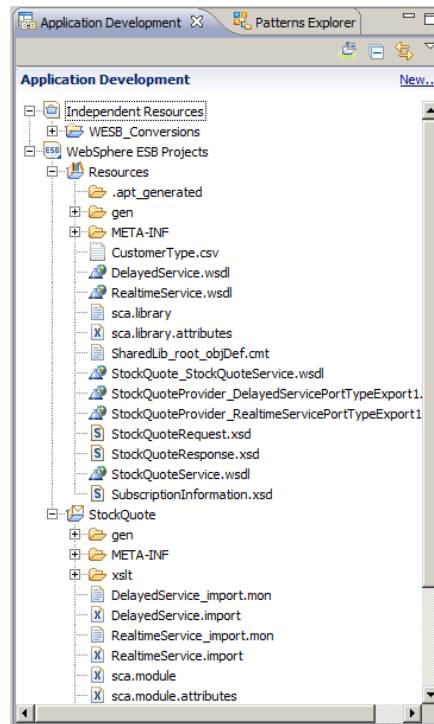
- A “New WebSphere ESB Conversion Session” will open.
Set “Conversion session filename” to **WESBConvSession1**.



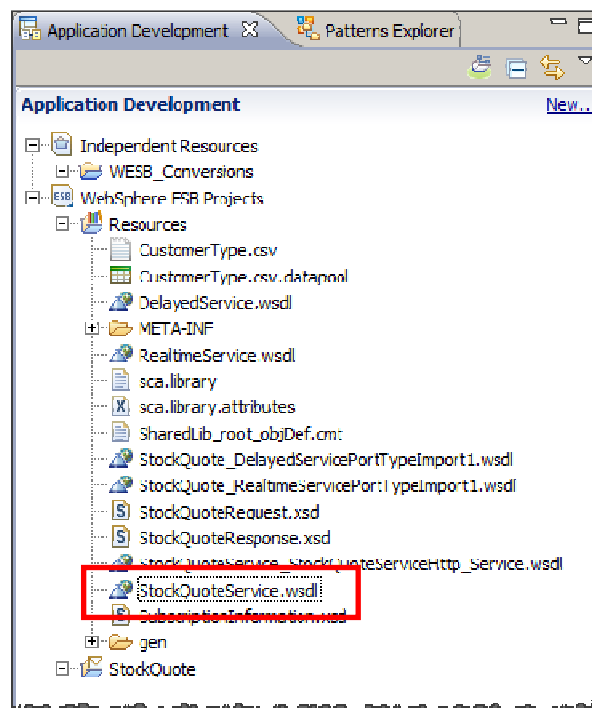
Click Finish.

5. Go to the “Application Development” view, expand the “Resources” and “StockQuote” folders under “**WebSphere ESB Projects**”.

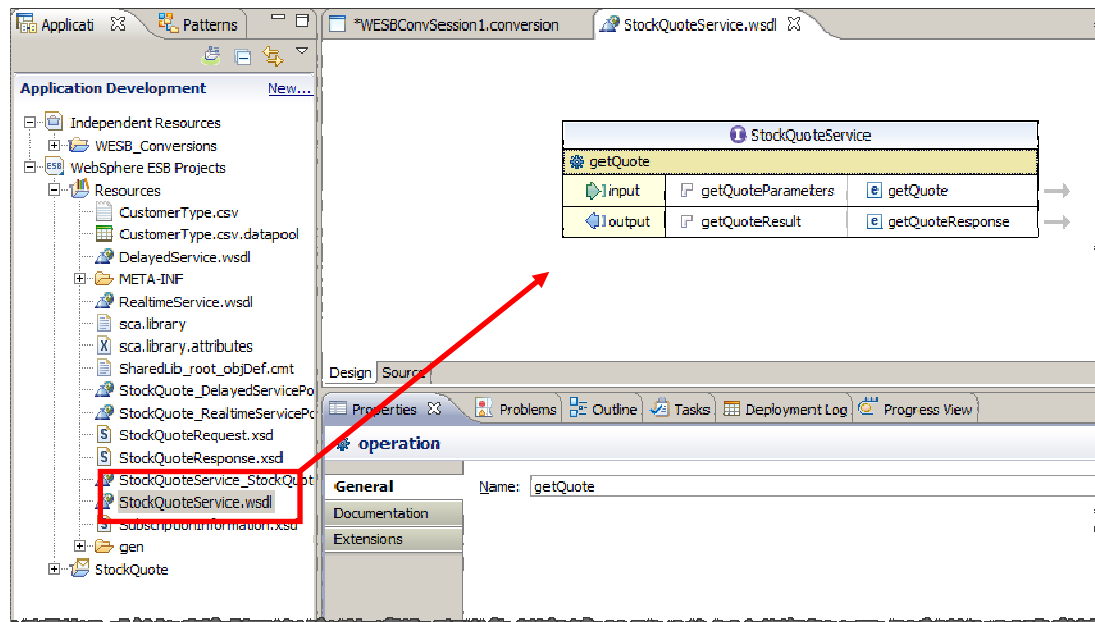
These are the components of the original StockQuote application, which will be converted by the Conversion tools.



6. In the Resources folder, open the StockQuoteService.wsdl file.

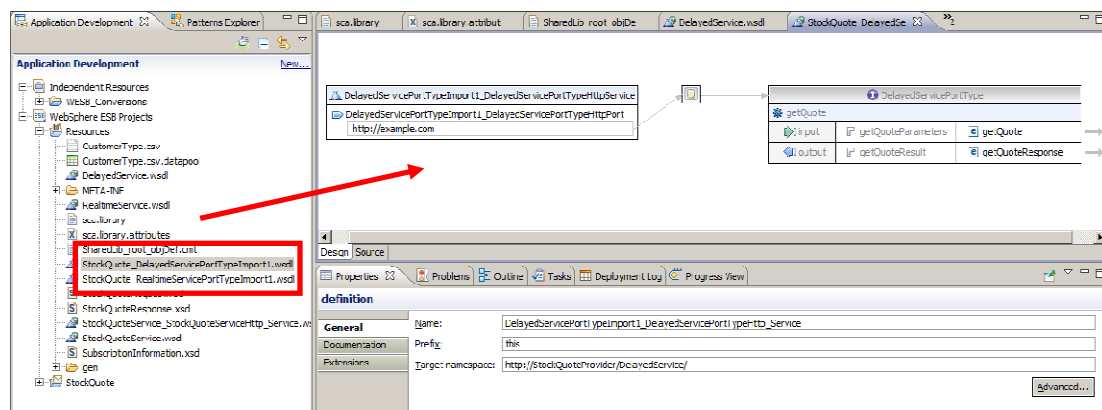


7. You can see the original WSDL design for the StockQuoteService in the editor. Do not modify this. Close the **StockQuoteService.wsdl** tab.



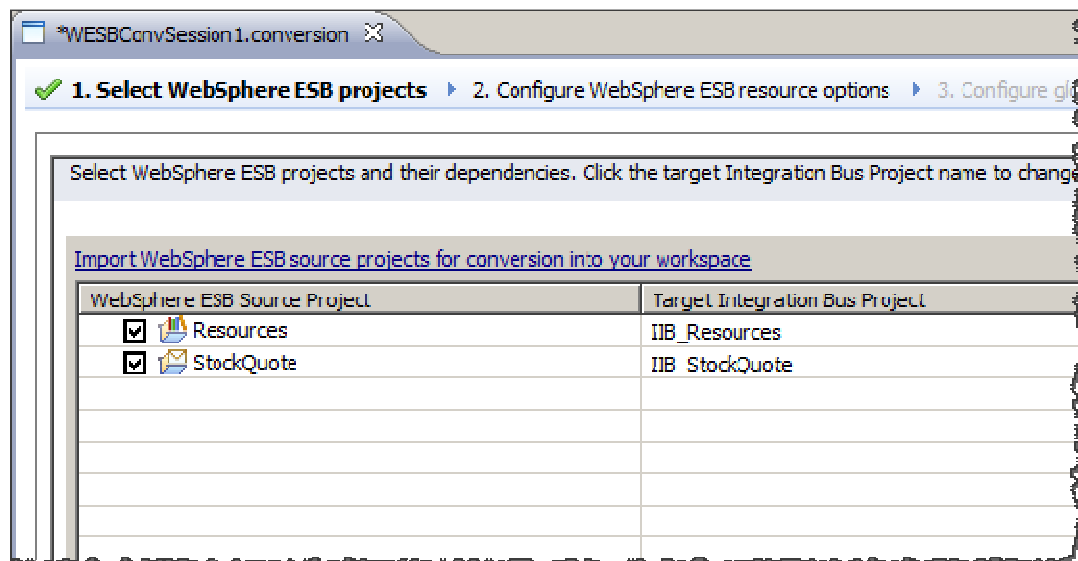
8. In the Resources folder, open the **"StockQuote_DelayedServicePortTypeImport1.wsdl"** file.

You can see the interface binding defined on the WESB project.

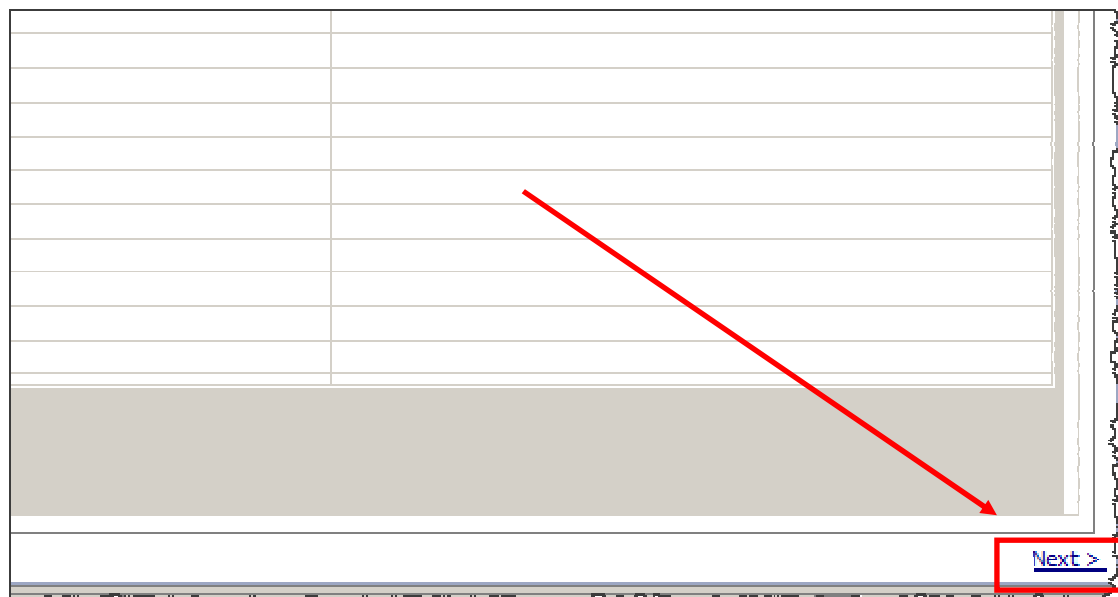


Close the wsdl without modifying.

9. Return to the editor for WESB conversion (WESBConvSession1.conversion). Both imported resources are going to be used for this conversion session, so make sure **“Resources”** and **“StockQuote”** are selected.



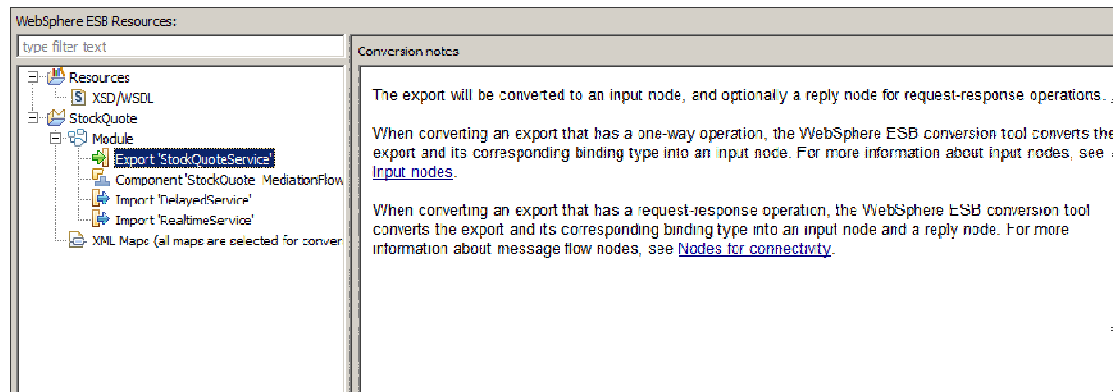
10. Click the “Next >” link to continue:



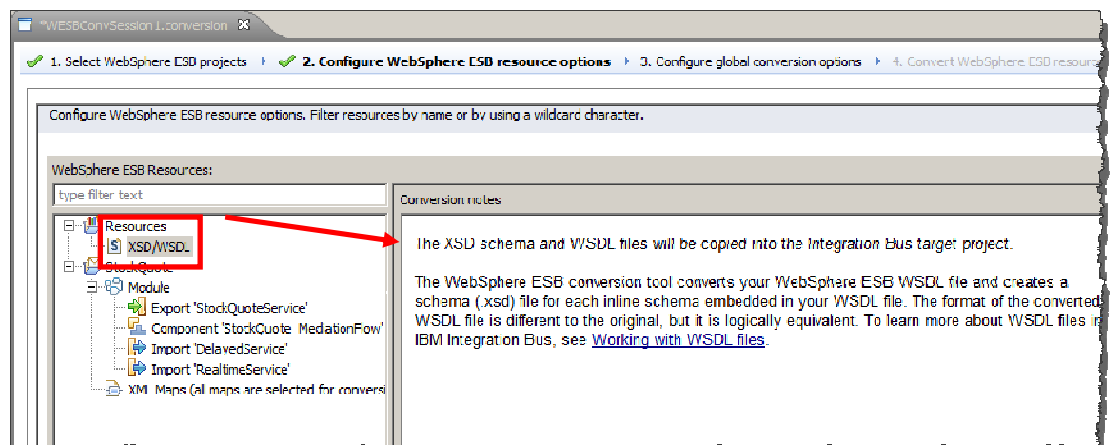
11. Click **“Export StockQuoteService”**. You will see the “Conversion notes” for this element; the notes describe the conversion that is assigned for this item.

For example, in the StockQuoteService item, the component is the initial service described in the original StockQuote application. It will be converted to an input node (SOAP/HTTP) with a SOAP Reply node that will be generated for request-respond scenarios.

Selecting other items will display information about the proposed conversion for that item.



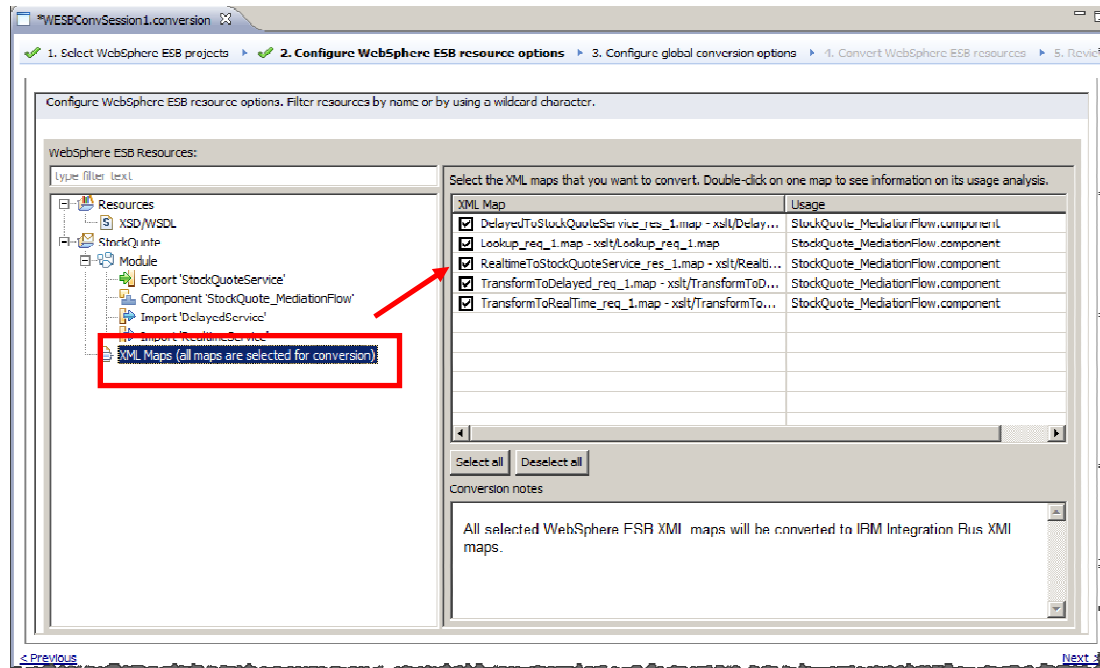
12. Now select Resources -> **XSD/WSDL**. The WSDL and XSD schema files represent the logical model of the message and define request calls/endpoints services for IIB:



13. Under the StockQuote item, select **“XML Maps (all maps are selected for conversion)”**.

All mappings will be selected and used for conversion. The converted mapping will be part of the resulting integration service, using the Integration Bus Mapping Node.

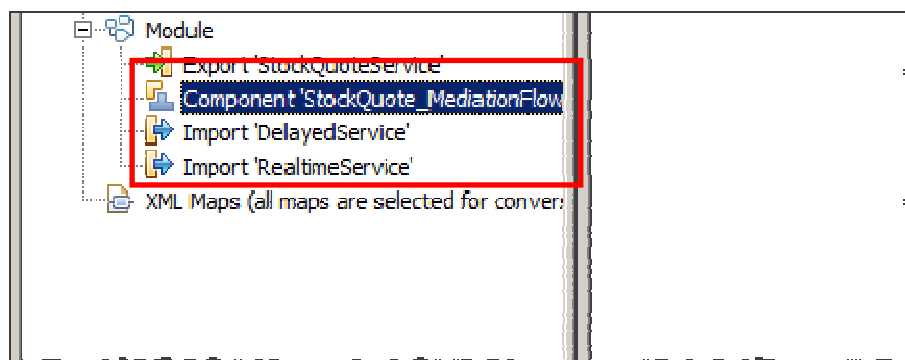
All transformations are converted, so you cannot deselect any of the map conversions.



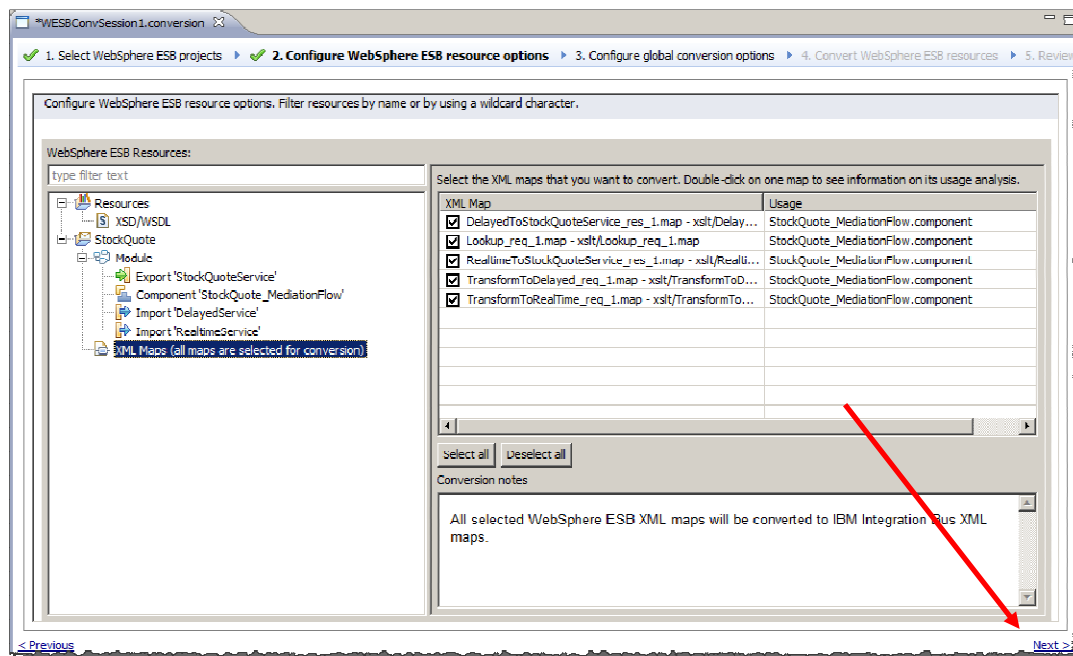
14. Similarly, review the “Conversion Notes” for:

StockQuote:

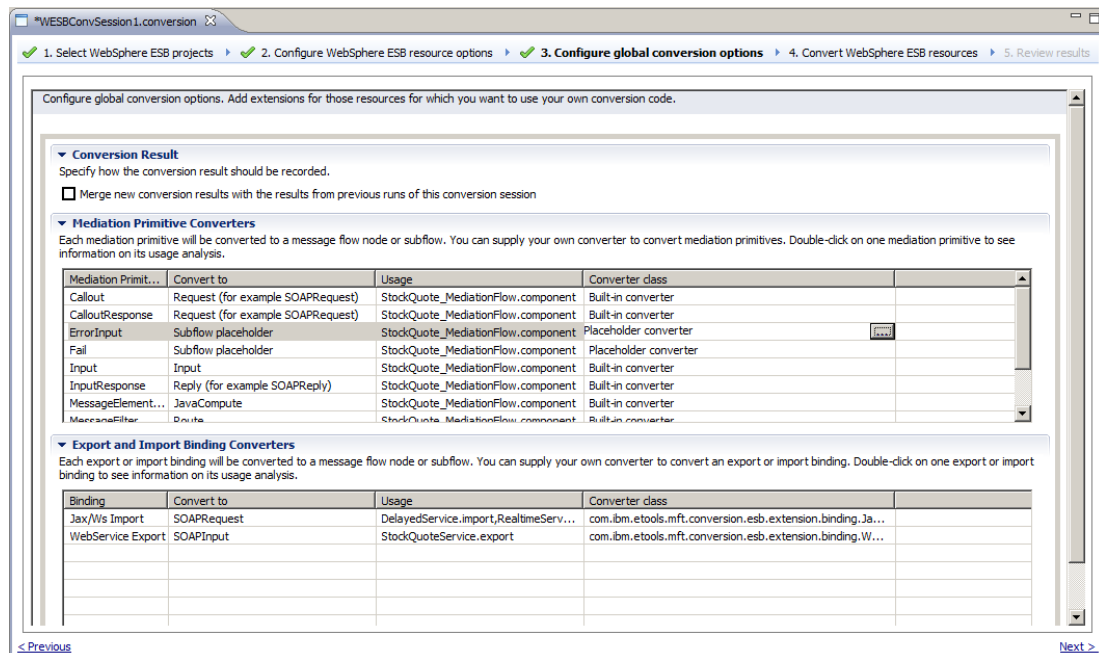
- Module
- **Component ‘StockQuote_MediationFlow’**
- **Import ‘DelayedService’**
- **Import ‘RealtimeService’**



15. Click “Next >” to continue with the conversion:



16. In this section you will find a summarized view of the original component and the conversion destination component:



17. Each mediation primitive will be converted to a message flow node or subflow. You can also create a new converter class, which allows you to create custom nodes or subflows for each primitive.

▼ **Mediation Primitive Converters**

Each mediation primitive will be converted to a message flow node or subflow. You can supply your own converter to convert mediation primitive to see information on its usage analysis.

Mediation Primit...	Convert to	Usage	Converter class
Callout	Request (for example SOAPRequest)	StockQuote_MediationFlow.component	Built-in converter
CalloutResponse	Request (for example SOAPRequest)	StockQuote_MediationFlow.component	Built-in converter
ErrorInput	Subflow placeholder	StockQuote_MediationFlow.component	Placeholder converter
Fail	Subflow placeholder	StockQuote_MediationFlow.component	Placeholder converter
Input	Input	StockQuote_MediationFlow.component	Built-in converter
InputResponse	Reply (for example SOAPReply)	StockQuote_MediationFlow.component	Built-in converter
MessageElement...	JavaCompute	StockQuote_MediationFlow.component	Built-in converter
MessageFilter	Route	StockQuote_MediationFlow.component	Built-in converter

▼ **Export and Import Binding Converters**

18. You will see several “Built-in converter” classes. These classes will convert the imported WESB item into the equivalent component in Integration Bus.

This means the converter will use prebuilt template classes to make those conversions.

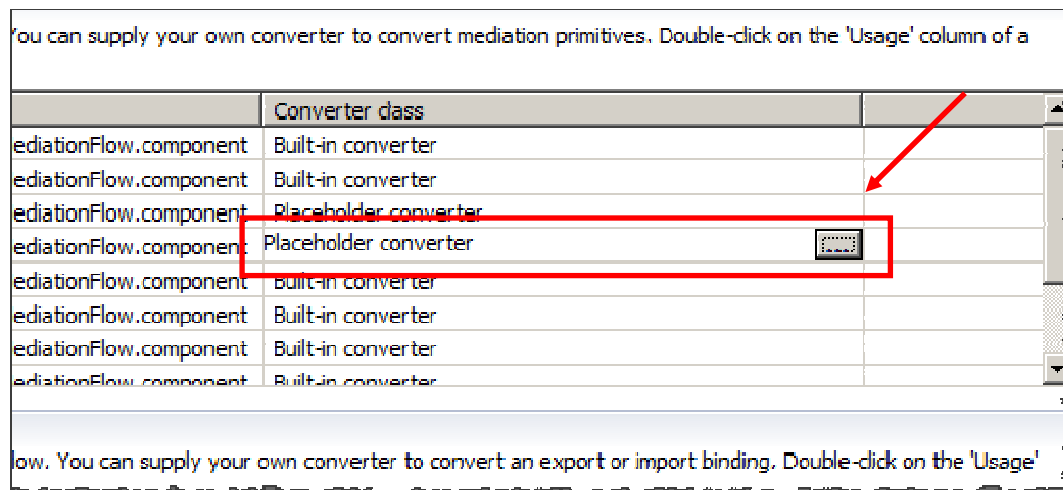
u can supply your own converter to convert mediation primitives. Doub

	Converter class
MediationFlow.component	Built-in converter
MediationFlow.component	Built-in converter
MediationFlow.component	Placeholder converter
MediationFlow.component	Placeholder converter
MediationFlow.component	Built-in converter
MediationFlow.component	Built-in converter
MediationFlow.component	Built-in converter
MediationFlow.component	Built-in converter

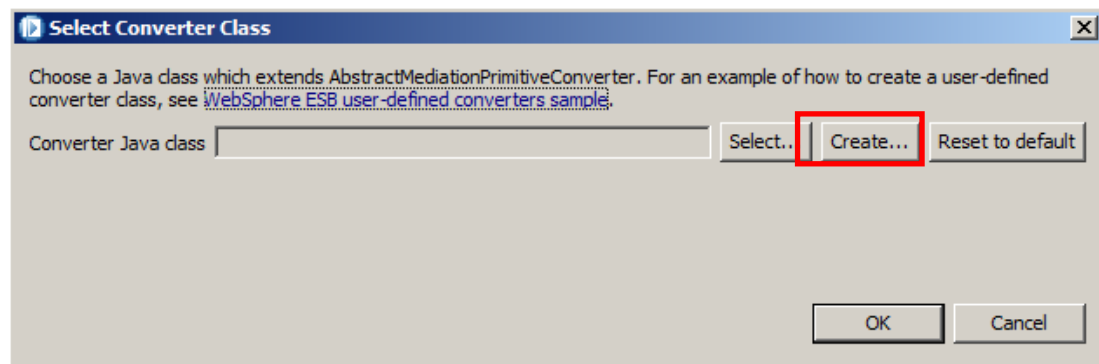
w. You can supply your own converter to convert an export or import

19. If an item has a **“Placeholder converter”**, then you can define your own conversion extension using the Java Integration API for IIB.

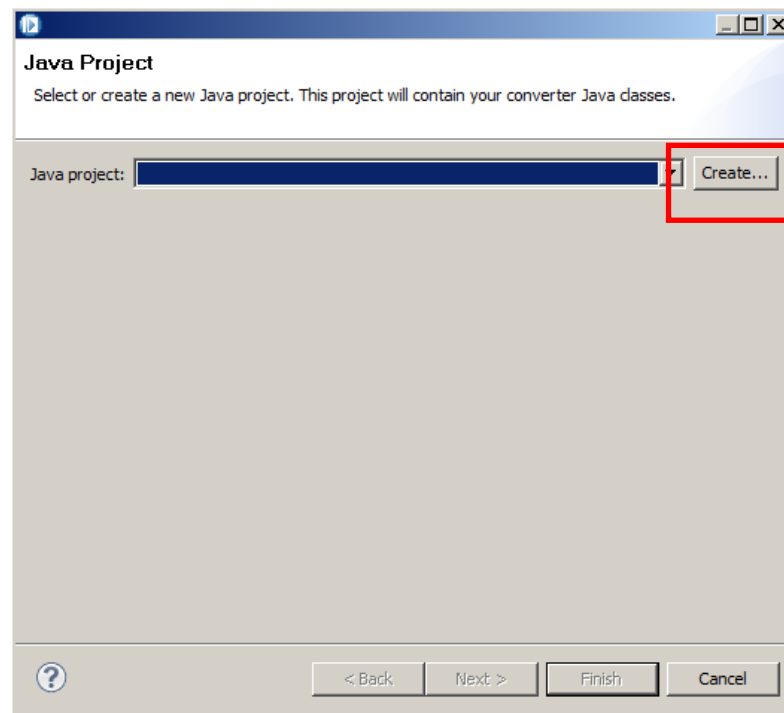
In this example, find the “MessageLogger” item. You will see the converter class is a “Placeholder converter”. Click (...) button on the “Placeholder converter” row to create a new converter Java class.



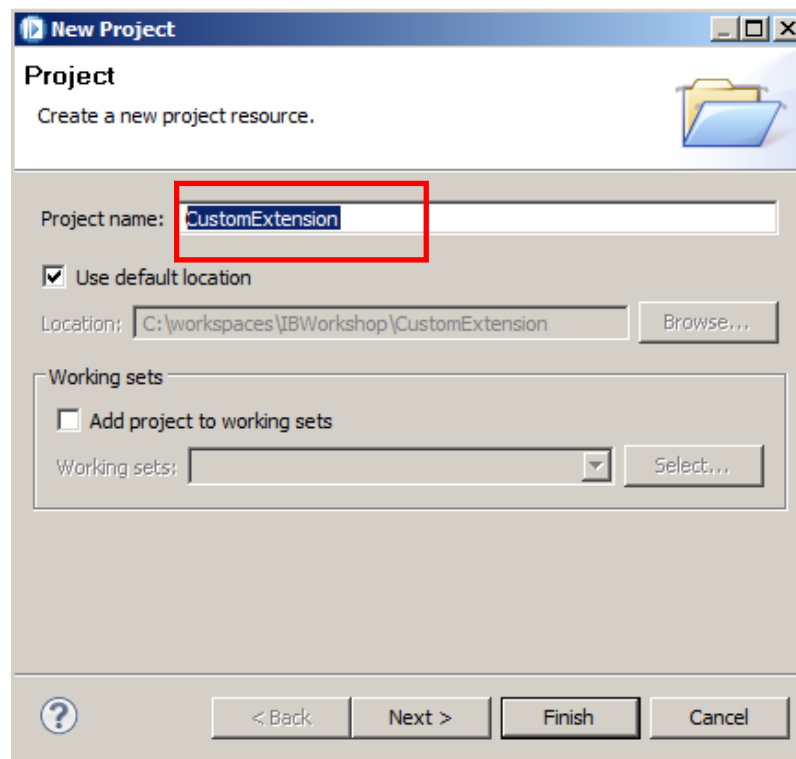
20. The Converter Class editor window will open. Click “Create...” to create a new extension class for this conversion:



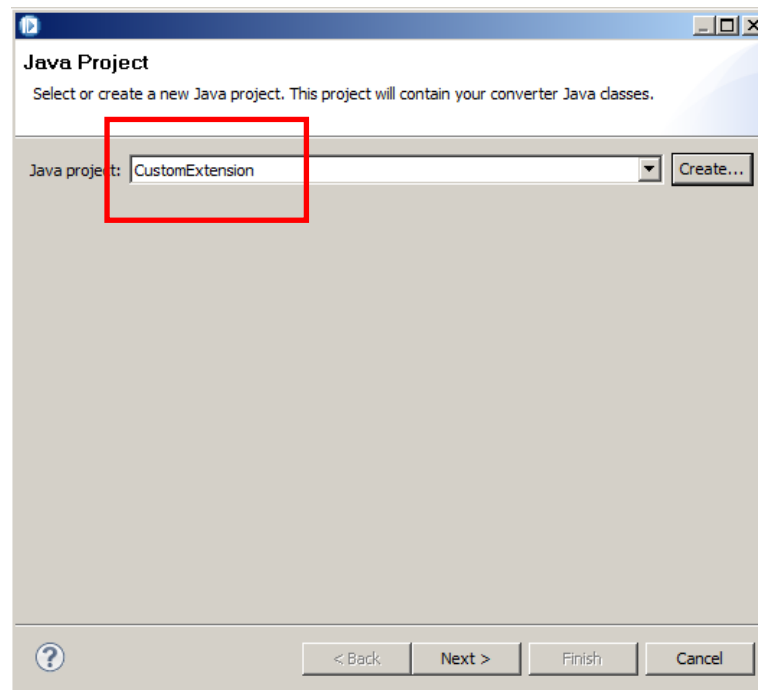
21. Click “Create...” button to create a new project to hold your new java class.



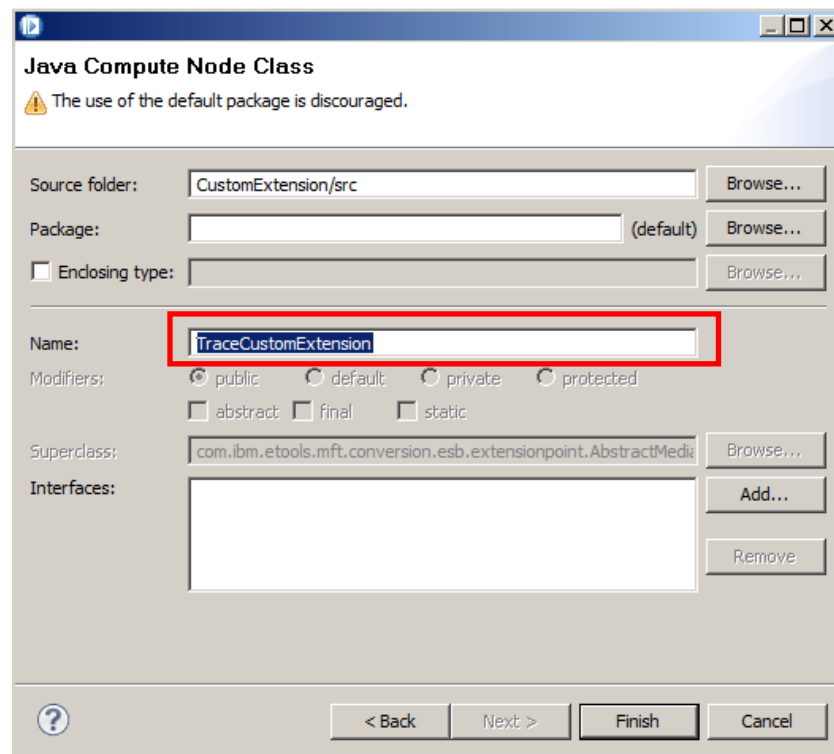
22. Set the Project Name to “CustomExtension”, and click Finish.



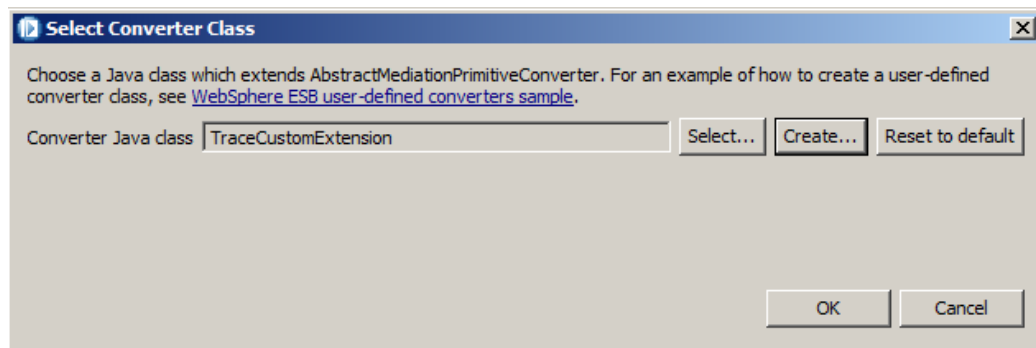
23. Click the “Next” button to create the class.



24. Set the class name to “TraceCustomExtension”. Ignore the warning about the default package. Click Finish.



25. Click OK to complete the creation of the new class.



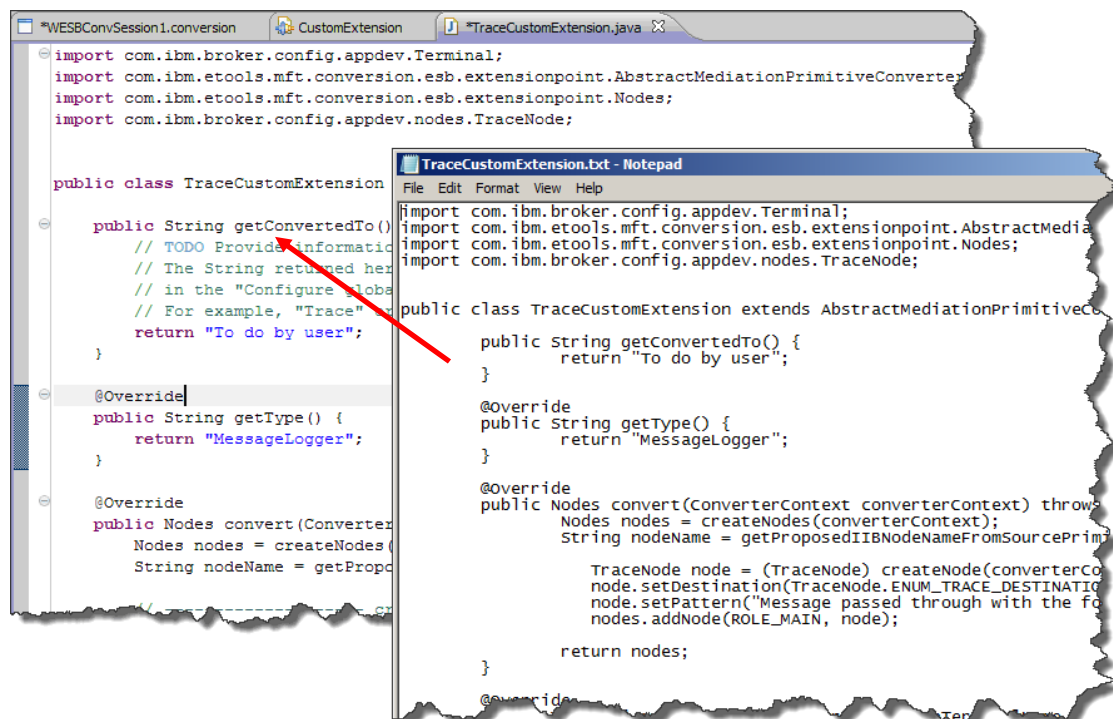
This will open a Java editor with the template Java code.

26. You will see a template class to create a flow using the Java API for IIB.

For this lab, we have provided the java code for you.

Open the file C:\student\WESB_Conversion\resources\TraceCustomExtension.txt.

Copy the entire contents of this file into the Java editor, ensuring that all the current code in the Java editor is deleted first.



27. The copied code uses the Integration API to create a new node in the converted flow. Review “**convert(ConverterContext converterContext)**” method of the class. This method creates a **TraceNode** for the logging of the transaction. Various node properties are also set with the java code. Take a few minutes to examine the java code, and observe how it creates, configures and connects the new node.

```
@Override
public Nodes convert(ConverterContext converterContext) throws Exception {
    Nodes nodes = createNodes(converterContext);
    String nodeName = getProposedIIBNodeNameFromSourcePrimitive(converterContext);

    TraceNode node = (TraceNode) createNode(converterContext.targetFlow, nodeName, ROLE_MAIN, TraceNode.class, nodes);
    node.setDestination(TraceNode.ENUM_TRACE_DESTINATION.localError);
    node.setPattern("Message passed through with the following fields: Bodycontent is ${Body} Date ${CURRENT_TIMESTAMP}");
    nodes.addNode(ROLE_MAIN, node);


    return nodes;
}
```

Save (Ctrl+s). Close the “CustomExtension” tab and “TraceCustomExtension.java” Java editor.

28. Return to “**WESBConvSession1.conversion**” tab.

Review the converter class for the “MessageLogger” mediation primitive. It should now be set to “**TraceCustomExtension**”.

ply your own converter to convert mediation primitives. Double-click on the 'Usage' column of

	Converter class	
bw.component	Placeholder converter	
bw.component	Built-in converter	
bw.component	Built-in converter	
bw.component	Built-in converter	
bw.component	Built-in converter	
bw.component	TraceCustomExtension	
bw.component	Built-in converter	

an supply your own converter to convert an export or import binding. Double-click on the 'Usag

Click “Next” link to continue.

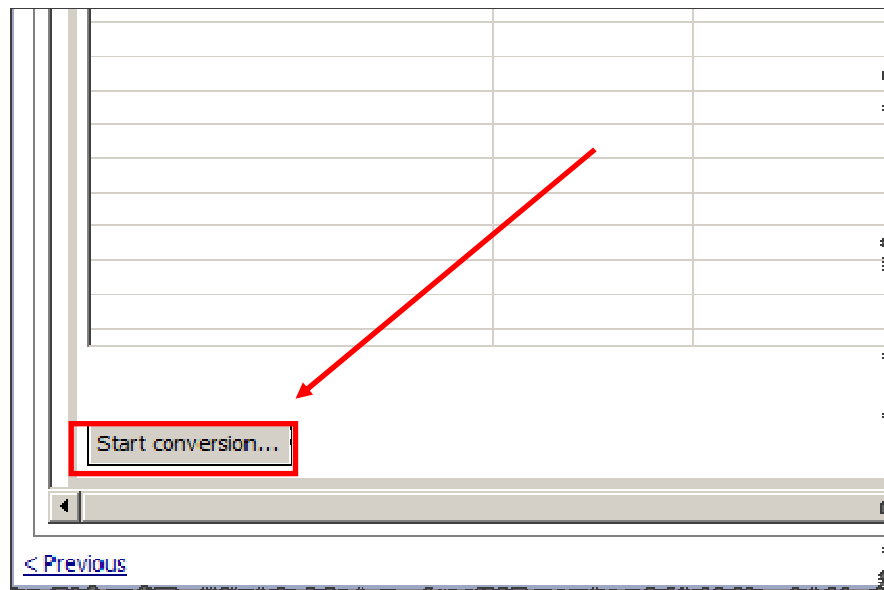
29. A summary for the conversion scenario will appear. Review the summary:

Review the configuration details for this conversion scenario. Start the conversion of your WebSphere ESB resources. Double-click on a map entry to verify the list of maps that are selected for conversion.

Summary of the conversion configuration:

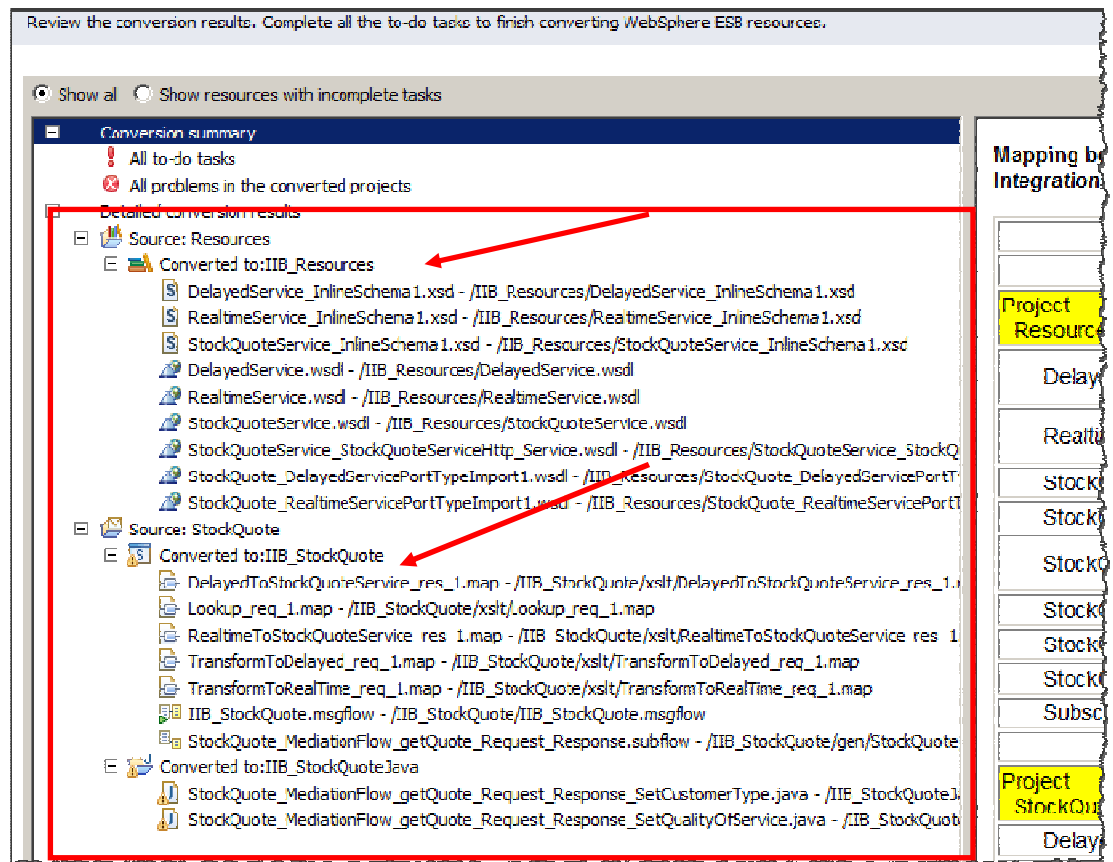
WebSphere ESB resource	Type	Option	Value
StockQuote	Project	XML Maps to convert	xslt/TransformToRealTime_req_1.map,xslt/TransformTo...
	Global option fil...	Merge conversion result	false
MessageLayout	Mediation primit...	Converter	TraceCustomExtension

30. Click “Start conversion...” in the bottom of the summary to start the conversion process:

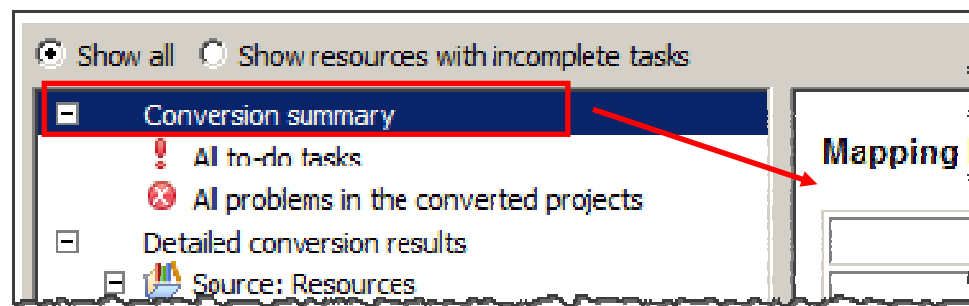


This process should take less than a minute.

31. After the conversion process is complete, review the conversion results and check the resulting IIB projects.



32. You can also see a mapping or equivalent table between the WebSphere ESB files and Integration Bus files, by clicking on the conversion summary item on the left panel:



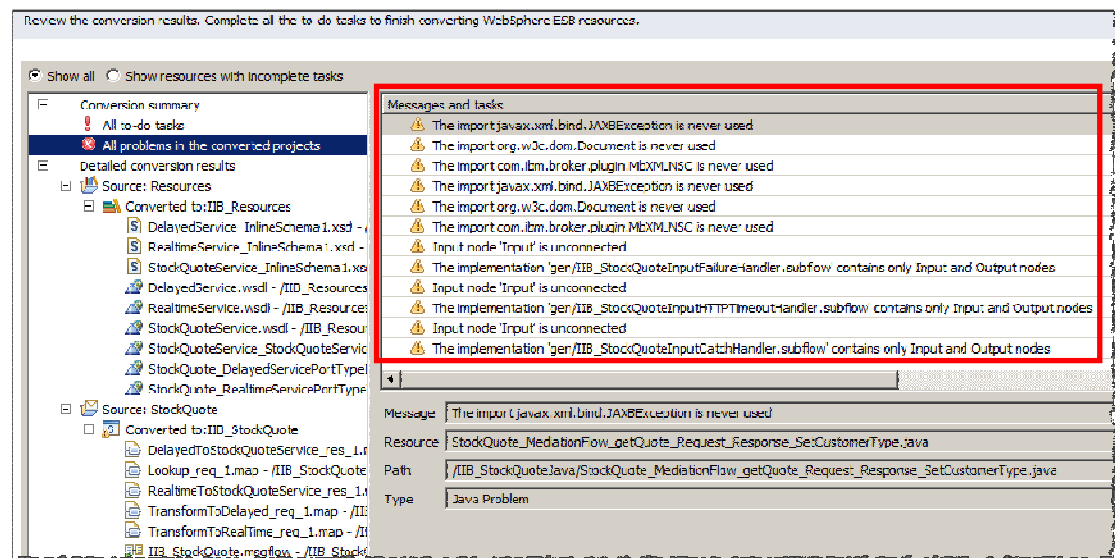
“Mapping between WebSphere ESB files and Integration Bus files”

Mapping between WebSphere ESB files and Integration Bus files

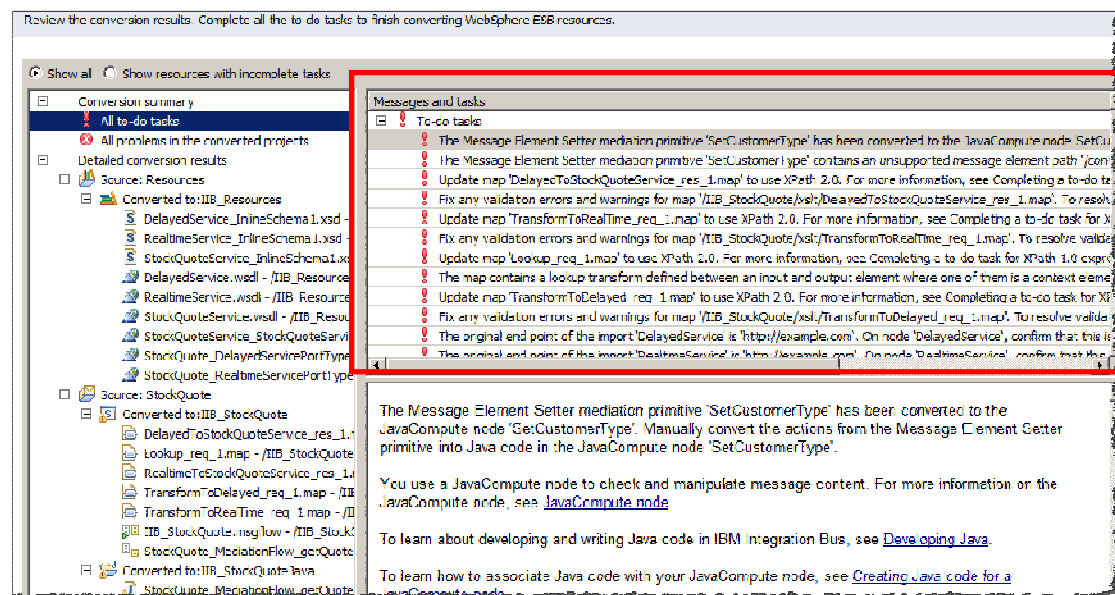
WebSphere ESB files	Integration Bus files
Project Resources	Project IIB_Resources
DelayedService.wsdl	DelayedService.wsdl DelayedService_InlineSchema1.xsd
RealtimeService.wsdl	RealtimeService.wsdl RealtimeService_InlineSchema1.xsd
StockQuoteRequest.xsd	StockQuoteRequest.xsd
StockQuoteResponse.xsd	StockQuoteResponse.xsd
StockQuoteService.wsdl	StockQuoteService.wsdl StockQuoteService_InlineSchema1.xsd
StockQuoteService_StockQuoteServiceHttp_Service.wsdl	StockQuoteService_StockQuoteServiceHttp_Service.wsdl
StockQuote_DelayedServicePortTypeImport1.wsdl	StockQuote_DelayedServicePortTypeImport1.wsdl
StockQuote_RealtimeServicePortTypeImport1.wsdl	StockQuote_RealtimeServicePortTypeImport1.wsdl
SubscriptionInformation.xsd	SubscriptionInformation.xsd
Project StockQuote	Project IIB_StockQuote
DelayedService.import	gen/StockQuote_MediationFlow_getQuote_request.subflow
RealtimeService.import	gen/StockQuote_MediationFlow_getQuote_request.subflow
StockQuoteService.export	IIB_StockQuote.msgflow
StockQuote_MediationFlow.component	StockQuote_MediationFlow_getQuote_request_SetCustomerType.java StockQuote_MediationFlow_getQuote_request_SetQualityOfService.java
StockQuote_MediationFlow.mfc	StockQuote_MediationFlow_getQuote_request/StockQuote_MediationFlow_getQu StockQuote_MediationFlow_getQuote_request/StockQuote_MediationFlow_getQu gen/StockQuote_MediationFlow_getQuote_request.subflow
sca.module	IIB_StockQuote.msgflow
xsit/DelayedToStockQuoteService_res_1.map	xsit/DelayedToStockQuoteService_res_1.map
xsit/Lookup_req_1.map	xsit/Lookup_req_1.map
xsit/RealtimeToStockQuoteService_res_1.map	xsit/RealtimeToStockQuoteService_res_1.map

33. Select “All problems in the converted projects”.

As you can see there are some warnings that you will fix in the next section of this lab.



34. Review the suggested tasks to solve some of these warnings by clicking “All to-do tasks”:



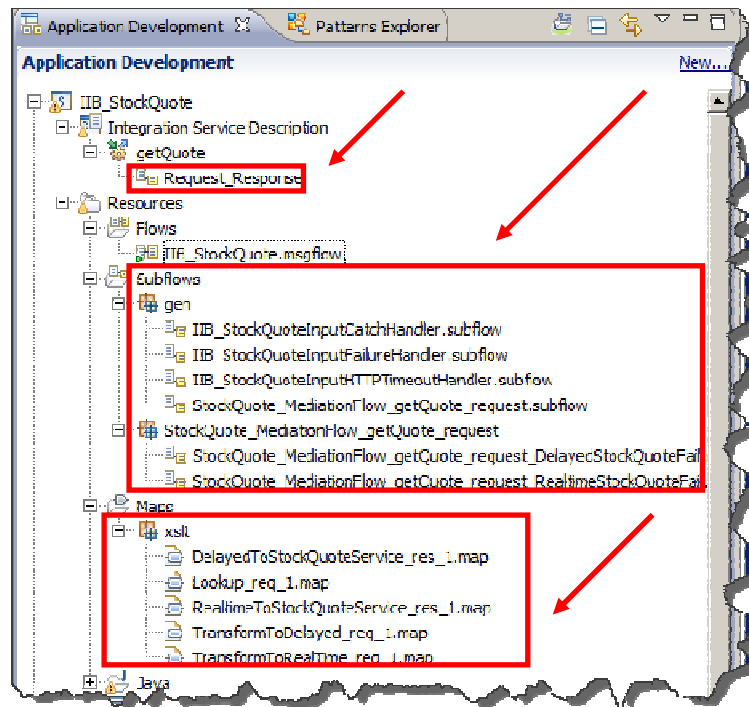
Save the “WESBConvSession1.conversion” (Ctrl+s) and close it.

3. Update the Generated StockQuote Service

1. Check the generated services and projects.

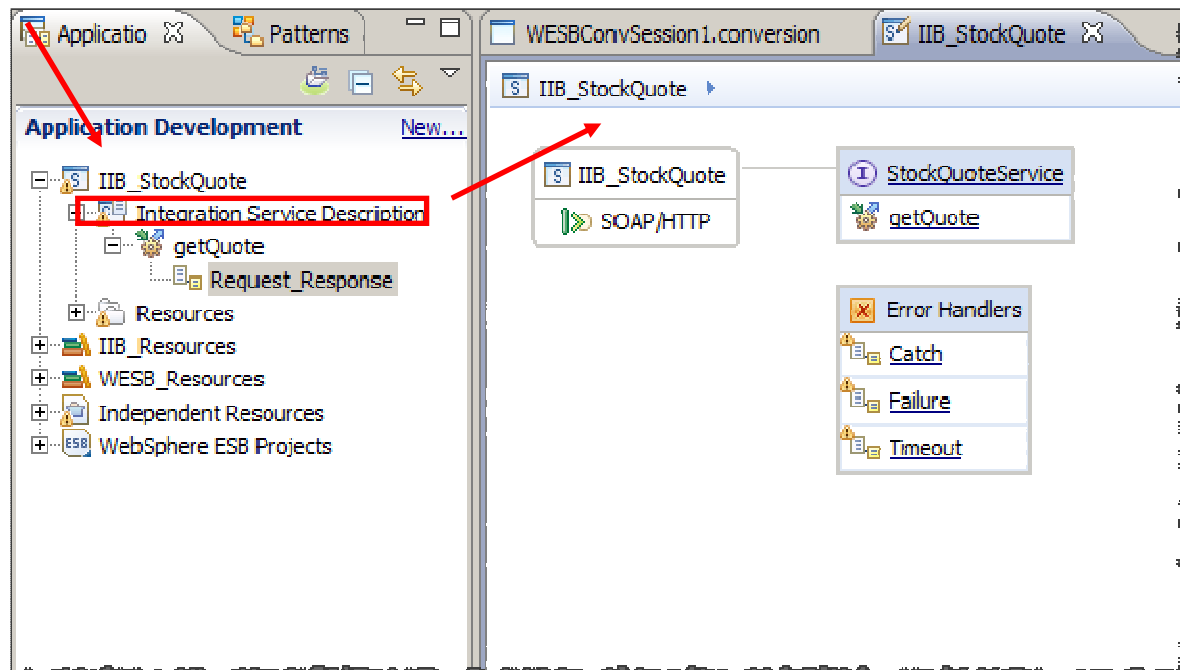
The first thing to note is that the conversion process has created an Integration Service called IIB_StockQuote. Expand the IIB_StockQuote Service and the IIB_Resources library.

In the IIB_StockQuote service, expand the Resources folder. You will see the generated Flows, subflows, mappings and other resources created by the conversion wizard:

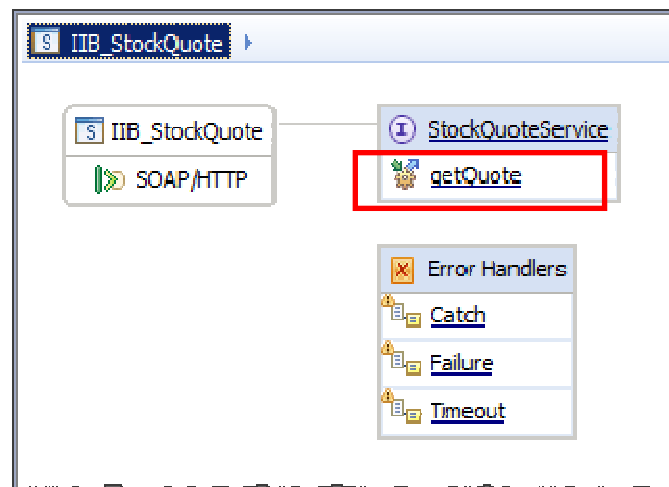


2. You will now perform some final conversion tasks that could not be done automatically.

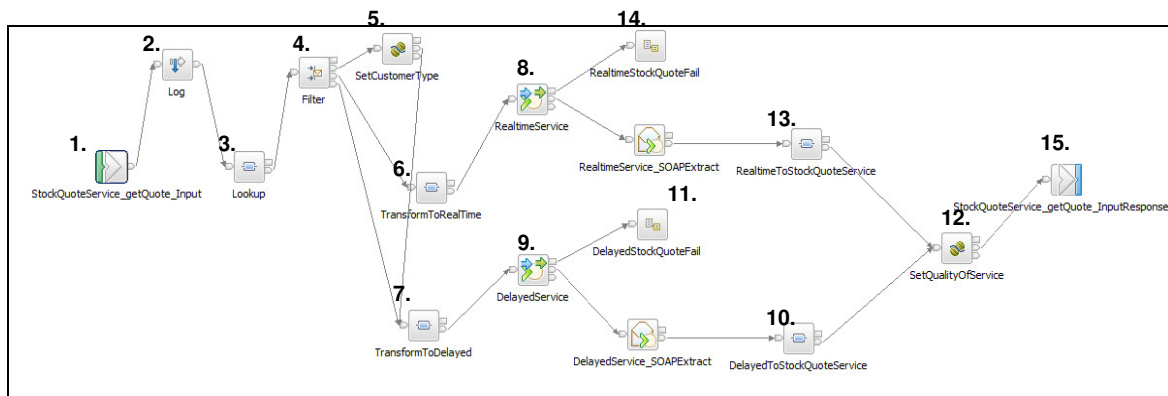
Open the generated Integration Service: "IBM_StockQuote" > **Integration Service Description**:



3. Click the "getQuote" link:

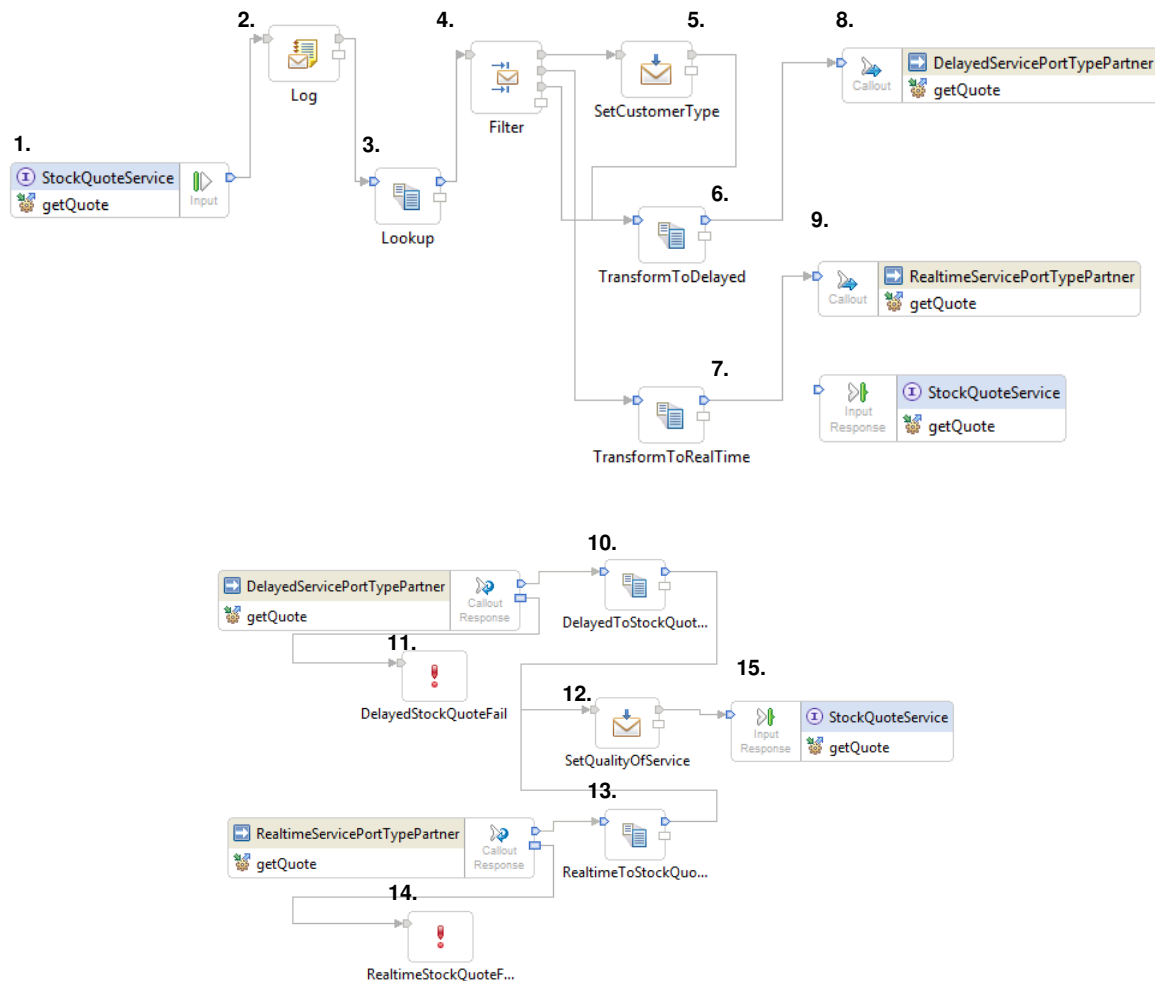


4. This will take you to the generated **Stock_MediationFlow_getQuote_request.subflow**. Note that the position of the node icons in your generated subflow may appear slightly different from those shown below.

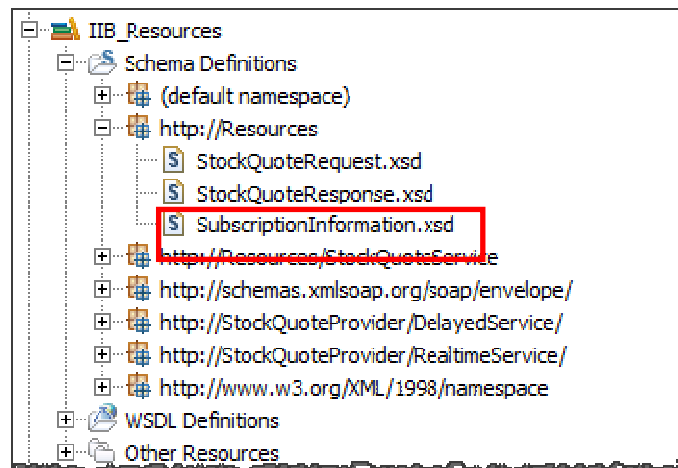


Note that some converted components have no native conversion equivalent so they have been converted to the most similar node available.

Actual StockQuote application sample mediation module for WESB:



5. Expand the “IIB_Resources” library. In Schema Definitions, expand “http://Resources”, and open **SubscriptionInformation.xsd**.

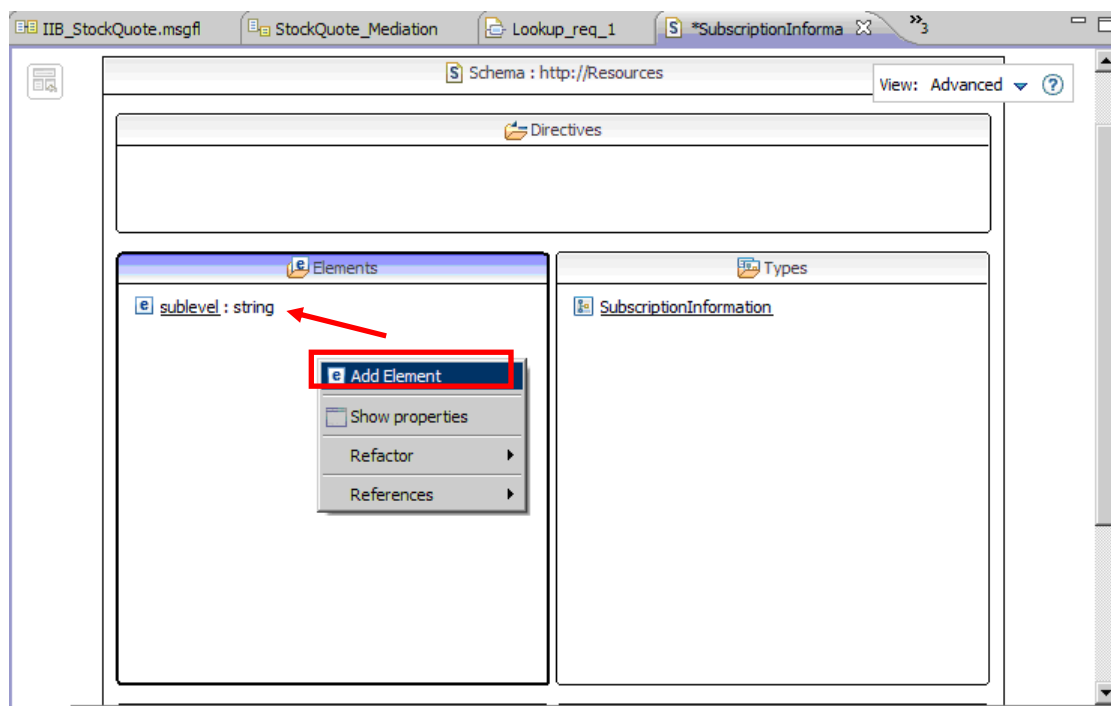


6. The SubscriptionInformation schema does not have any elements defined yet. This is because it was not exported as a complex business object. You are going to create a string element for the SubscriptionLevel schema.

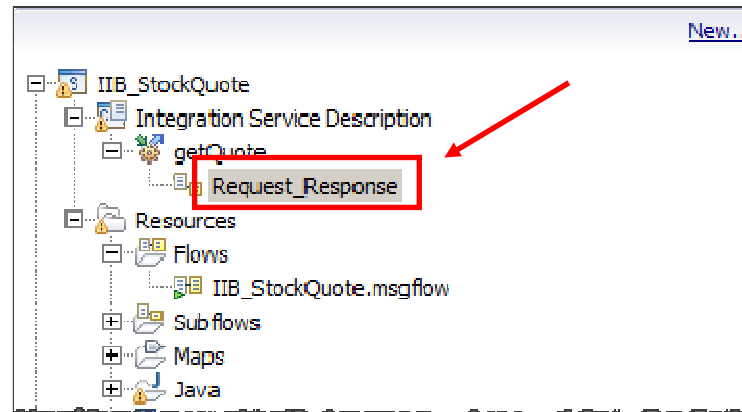
In the “Elements” white space, right-click and select “Add Element”. Name the new element “sublevel”.

This element will be used later with the LocalEnvironment to keep correlation information as the message passes through the flow. (Note, you can choose to use alternative methods to store correlation information; we are just using the Local Environment as a simple illustration of possible techniques).

Save the schema.

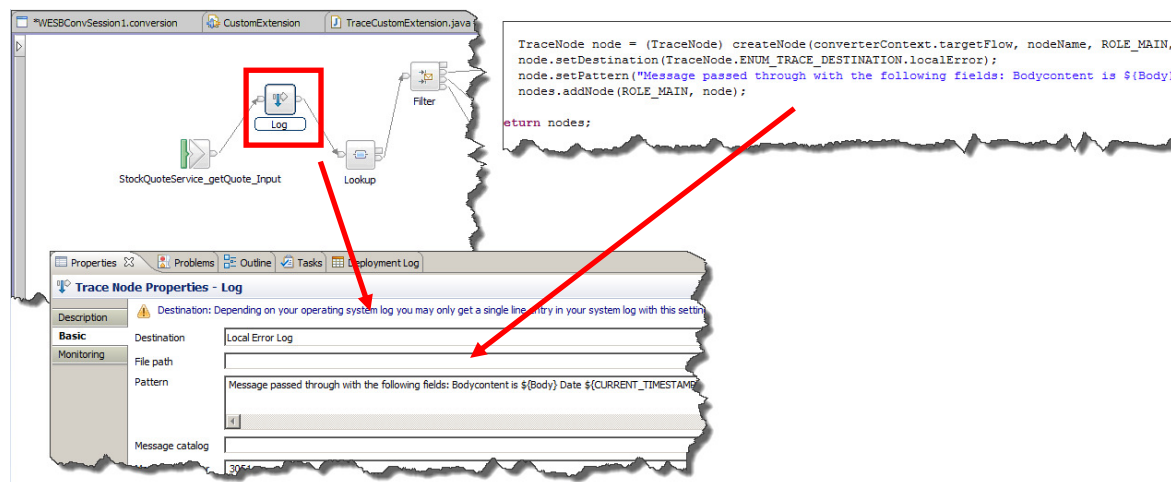


7. In the Application Development view go to IIB_StockQuote> Integration Service Description > getQuote > **Request_Response**. Open it.



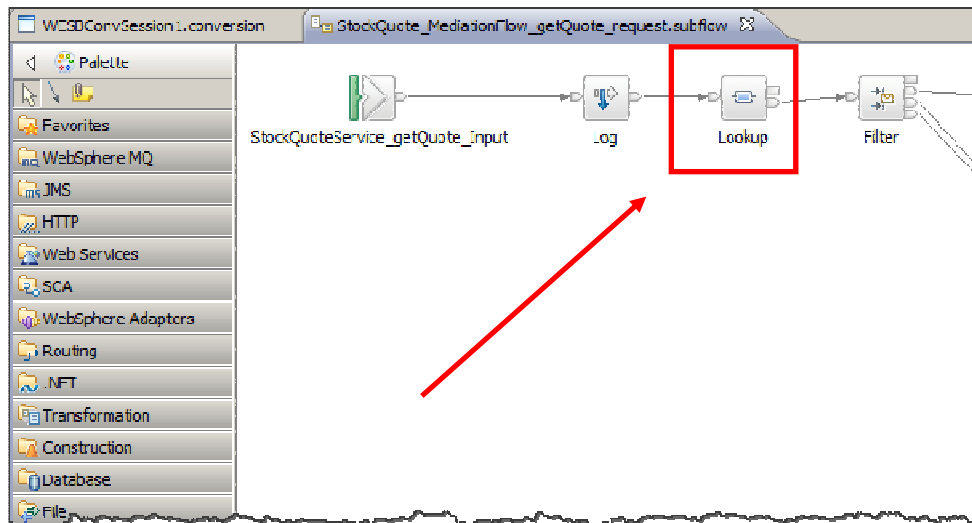
Remember this will take you to the **Stock_MediationFlow_getQuote_request.subflow** where you can find the actual mediation logic for this flow.

8. First, review the “Log” node (which is an Integration Bus Trace Node). This is the node that was generated by the Customer Extension java code, using the Integration API.



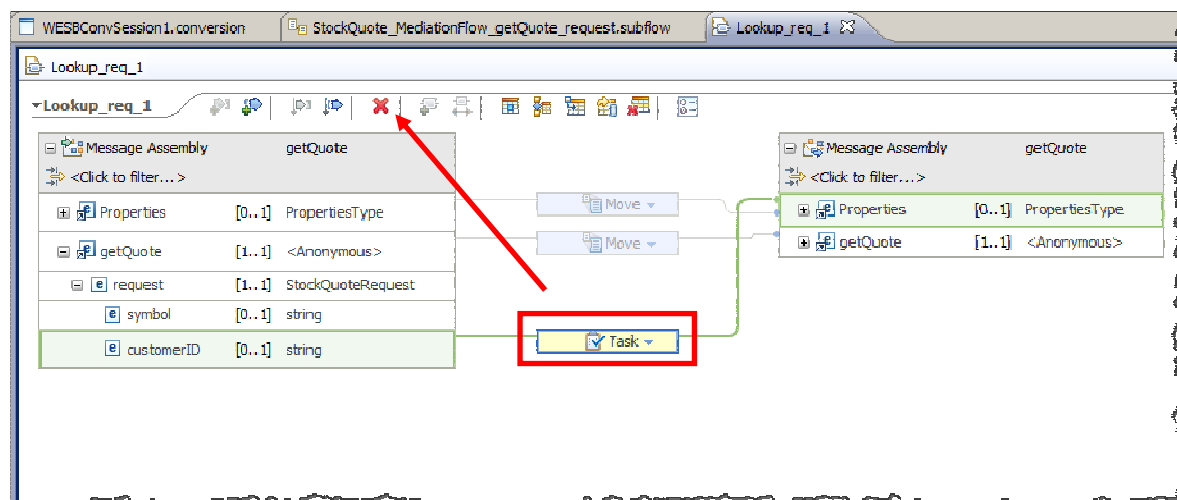
9. In the next few steps you will work with the Mapping Node editor to complete the required mappings. Two primary tasks are required:
- Customize the Lookup Mapping Node to read a Customer ID from a relational database table. As it was done in the original mediation flow.
 - Several Mapping Nodes will be customized to introduce the use of the Local Environment. As discussed above the LocalEnvironment is used as a convenient local storage area for correlation information (the sublevel item that was added to the schema).

First, you will customize the Lookup Mapping Node to retrieve the customer subscription level information. Go to the “Lookup” node and double-click on the node to open it.



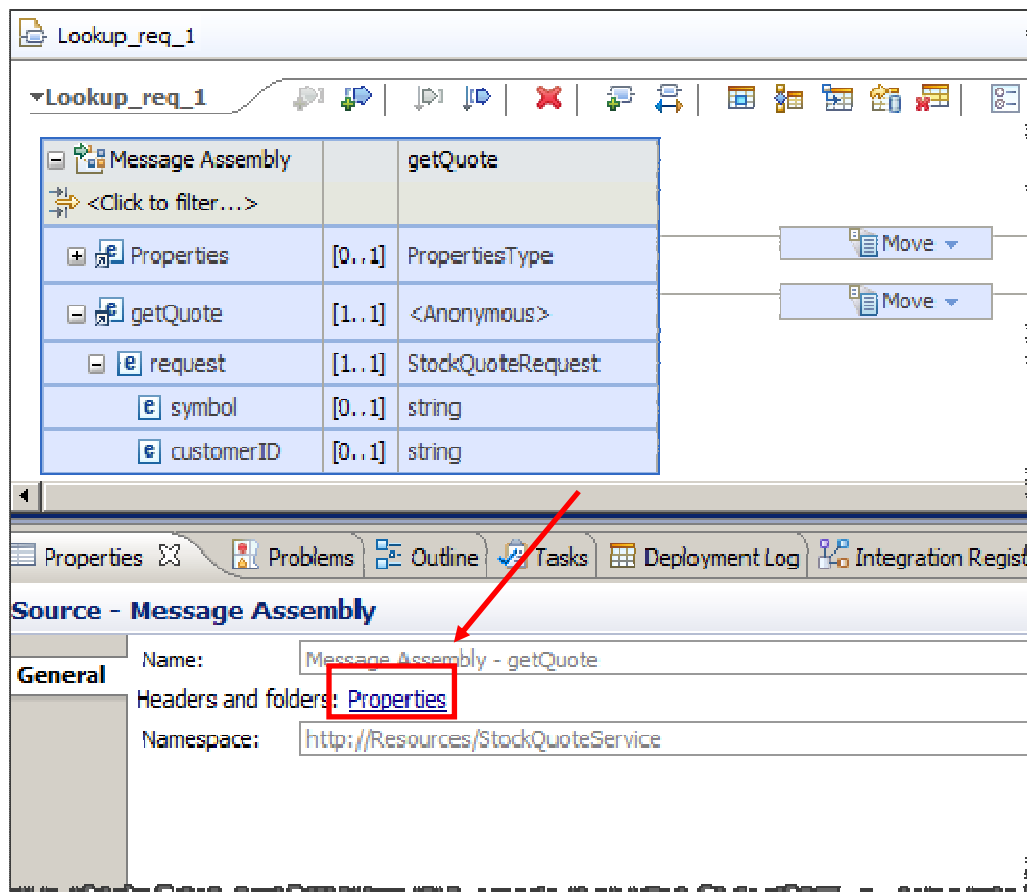
10. In the original flow, the database lookup node was been used to retrieve Customer information from a DataBase file. The mapping node has the ability to query the data source to add this data to any part of the message tree.

You will see a “Task” mapping, Delete the “Task” mapping, by right-clicking and delete or use the red “x” on the top.



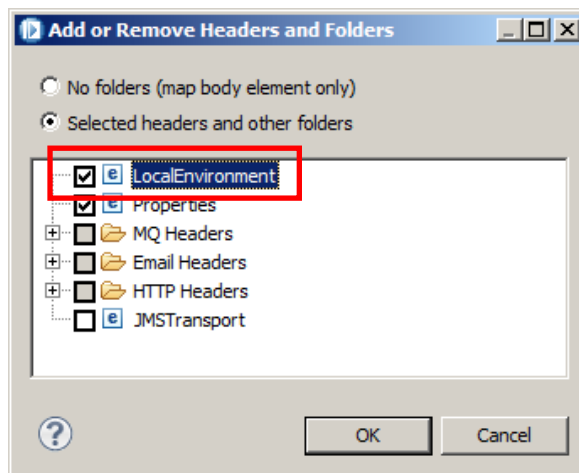
11. To get the subscription level of the Customer, the Mapping Node will query a table from the SAMPLE database, and store that information on the “LocalEnvironment” until it is needed again.

To do this, you need to enable the LocalEnvironment to be accessed by the Mapping Node. Select the “Message Assembly” box of the **input** object:



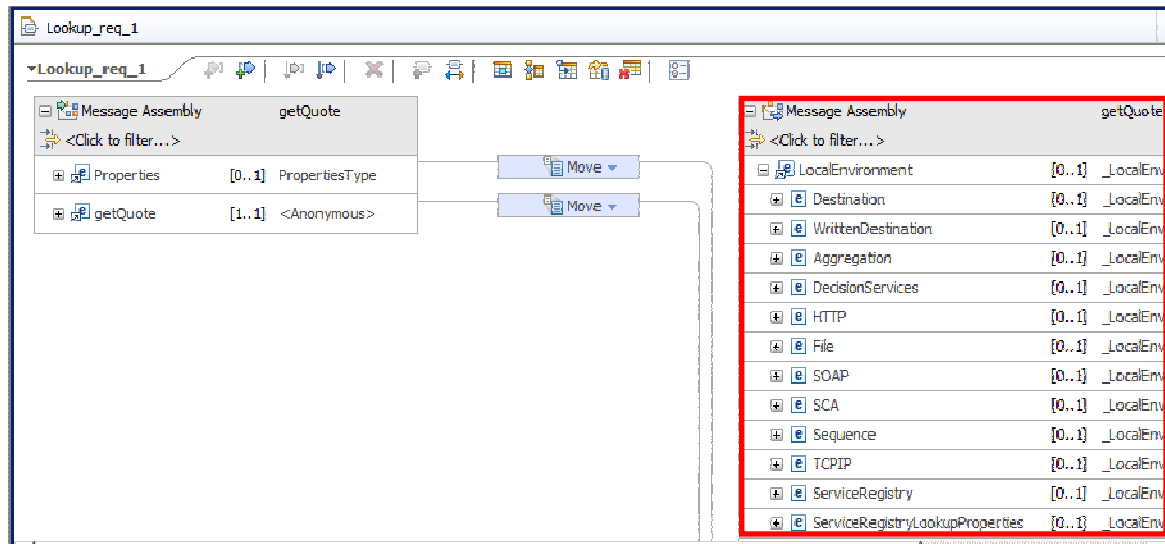
Click the link “Properties” on the “Properties” tab.

12. Mark the “LocalEnvironment” checkbox and click “OK”.

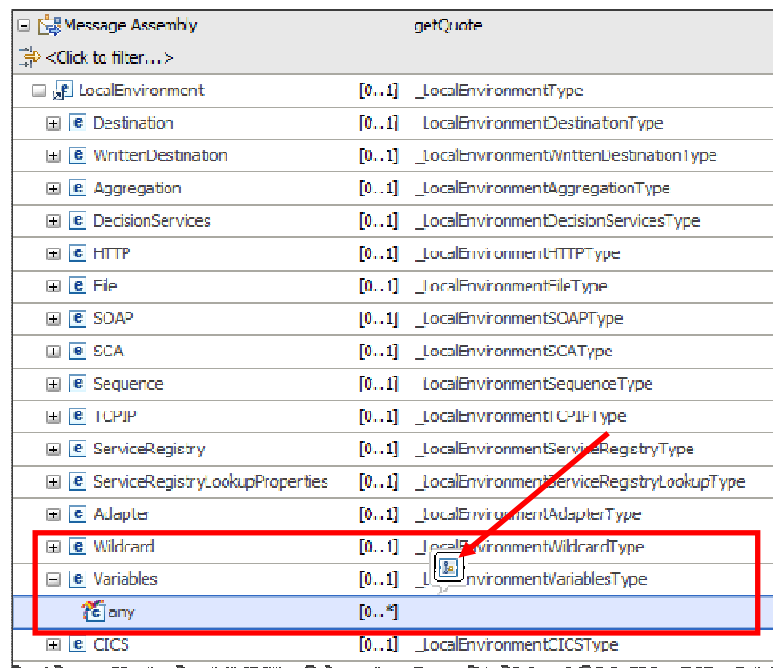


13. Perform the same procedure for the **output** message.

Now you will have a “LocalEnvironment” header for **output Message Assembly**. You will use this header to store the Subscription Level of the Customer:

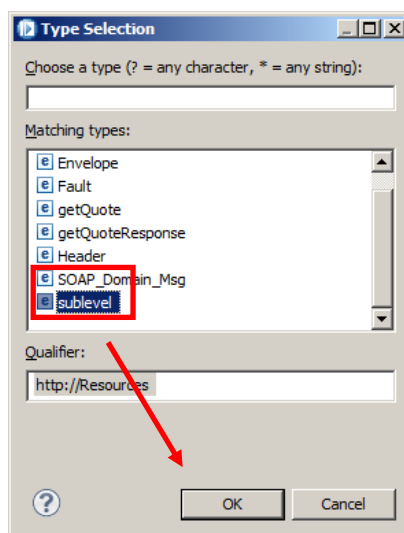


14. Create a variable in the LocalEnvironment to map the “sublevel” element previously defined. Expand “LocalEnvironment” on the output Assembly Message, expand Variables, and expand it to cast a new element:



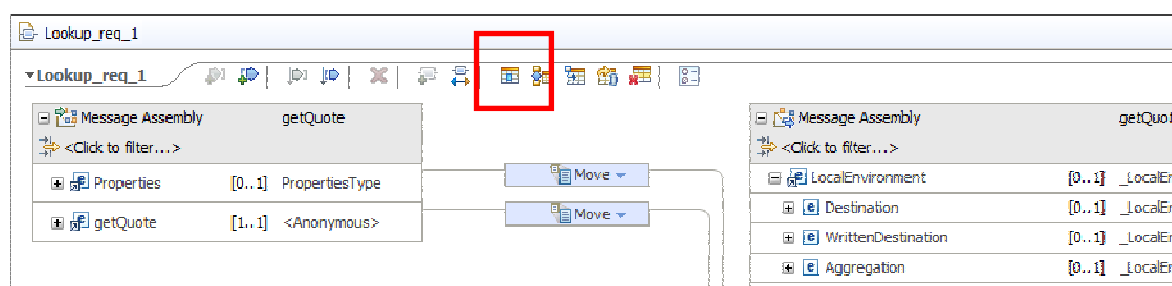
Click “any” and the “Cast” icon will appear. Click the Cast icon. (You can also right-click on the “any” line, and select cast from the Context Menu if the hover icon does not appear).

15. Select “sublevel” and click “OK”



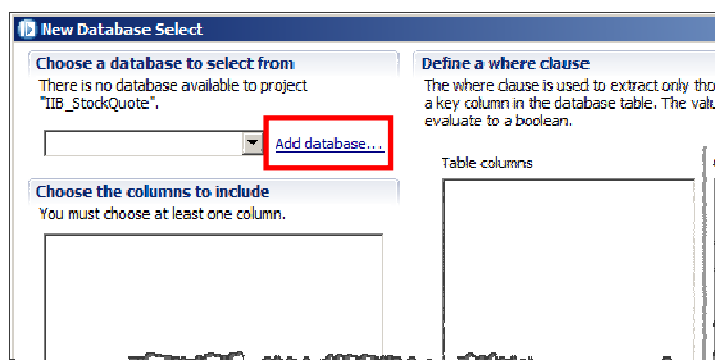
16. Now create a select statement to retrieve the customer information from the SAMPLE database.

Click the “Select rows from a database” icon:

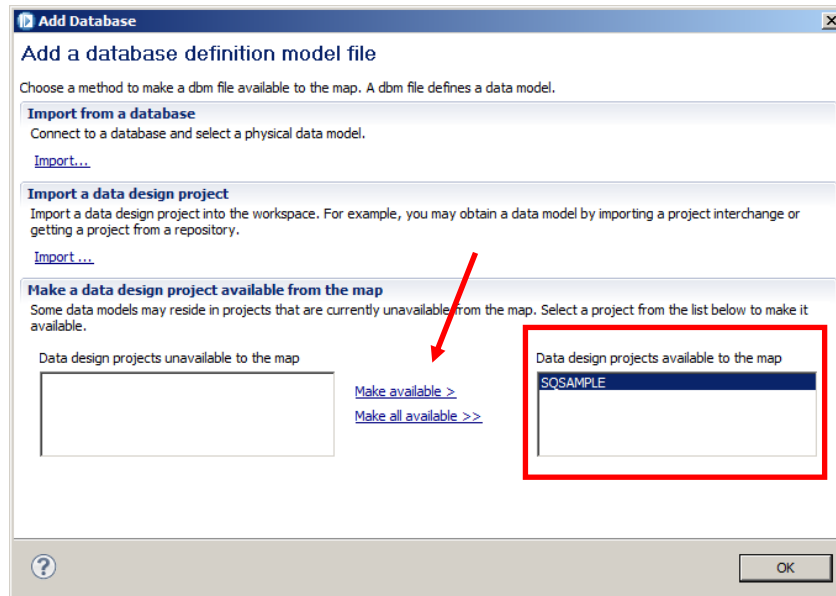


A “New Database Select” dialog will pop.

17. If there is no Database available in the dropdown menu of the “New Database Select” editor, you will need to add the database. Click on the “**Add database...**” link:



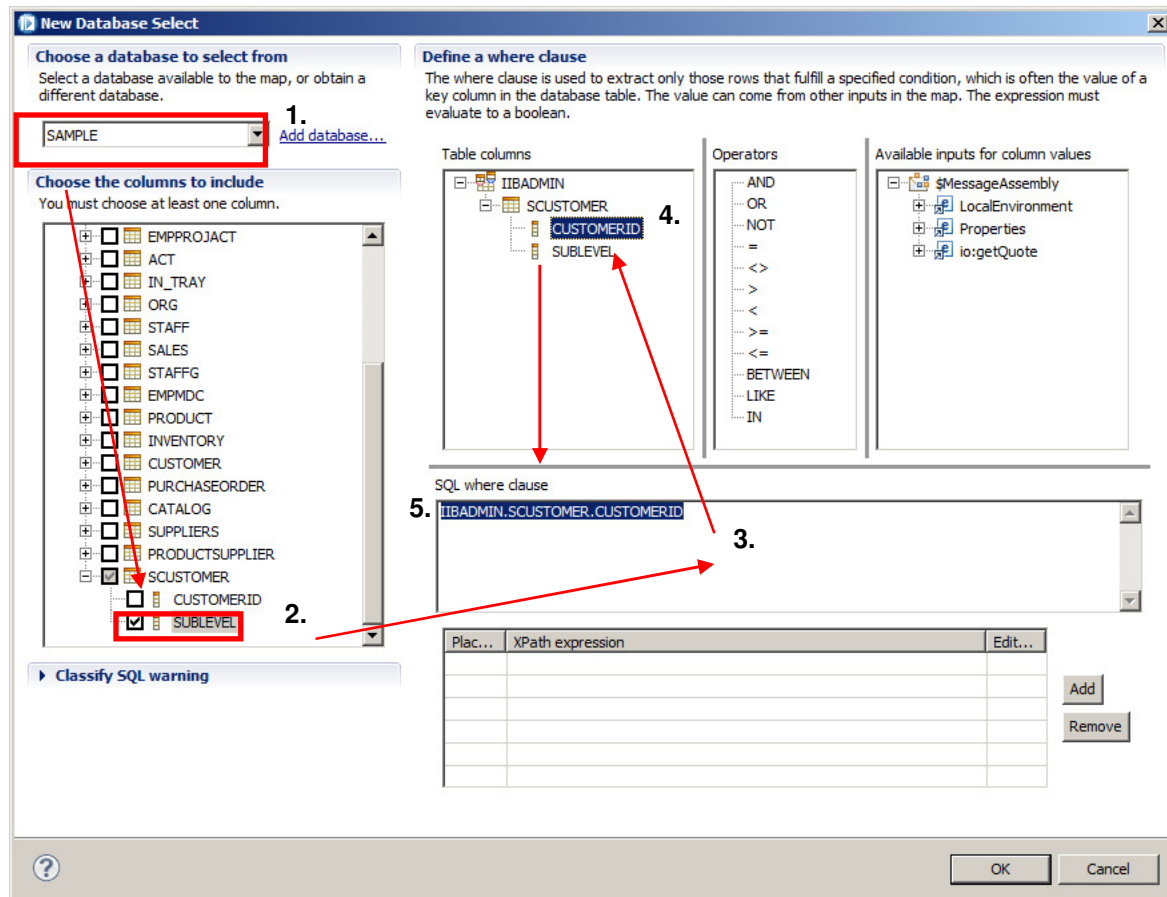
18. Select the SQSAMPLE database from the “Data design projects unavailable to the map” box and click “Make available”.



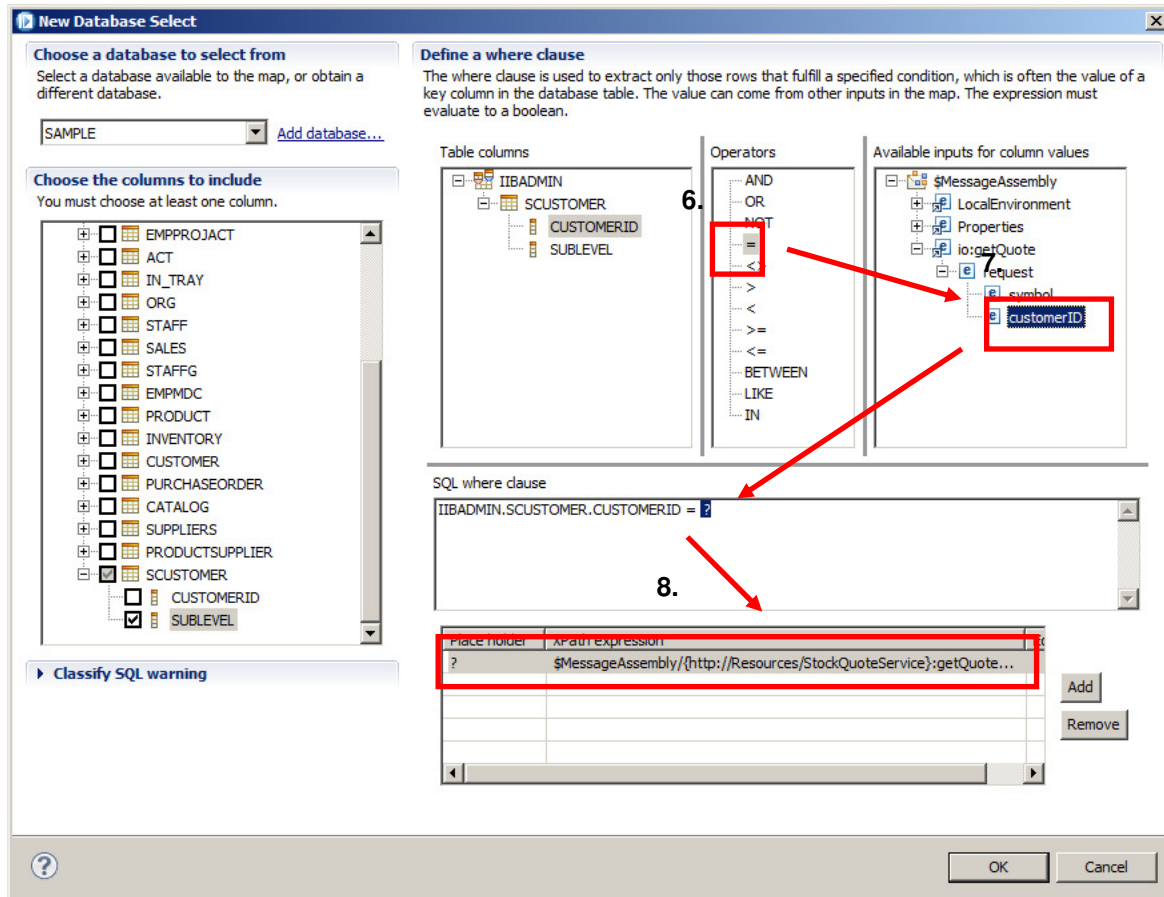
Make SQSAMPLE is now on the “Data design projects available to the map” box. Click “OK”

19. To build the SELECT expression for this task, complete the following steps:

1. Select the SAMPLE database from the drop-down menu.
2. Expand the SCUSTOMER table. Select only the "SUBLEVEL".
3. Delete the "1=1" default expression in the "SQL where clause" box.
4. Go to "Table columns". Expand IIBADMIN > SCUSTOMER. Double-click **CUSTOMERID**.
5. **Review** "SQL where clause" is "IIBADMIN.SCUSTOMER.CUSTOMERID".



20.
 6. Double-click on the “=” operator.
 7. Go to “Available inputs for column values” box, Expand io:getQuote > request. Double-click **customerID**.
 8. The “Place holder” that was generated for this expression refers to the customerID in the MessageAssembly.



Make sure the expression on the “SQL where clause” box looks like this

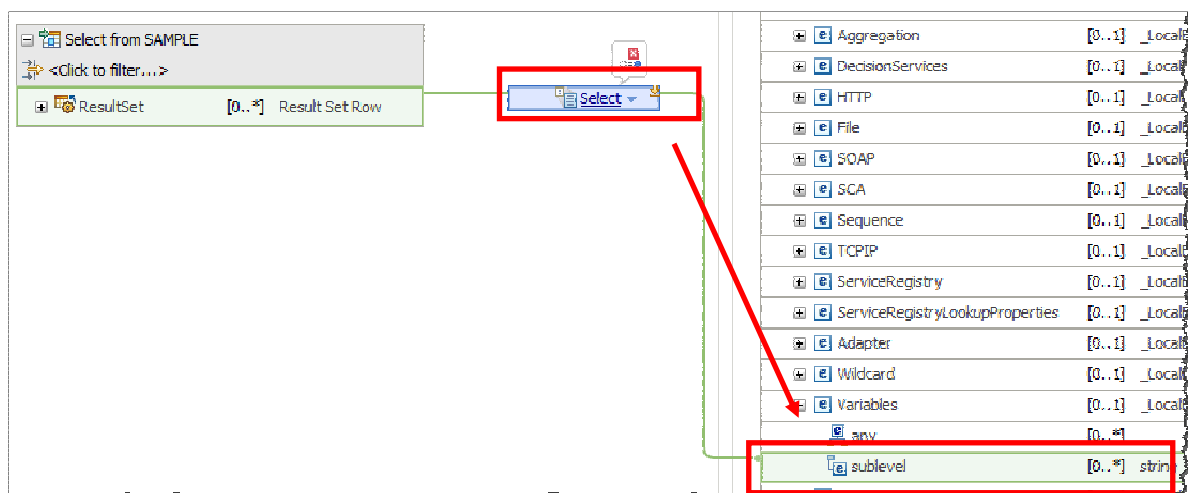
“IIBADMIN.SCUSTOMER.CUSTOMERID = ?”

Also make sure that the XPath Expression look like this:

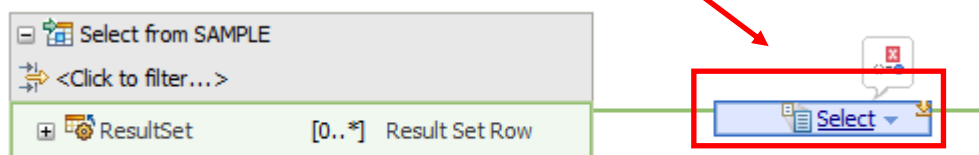
\$MessageAssembly/{http://Resources/StockQuoteService}:getQuote/request/customerID

Click OK.

21. Connect the “Select” mapping to with the “sublevel” Variable you created a few steps before:

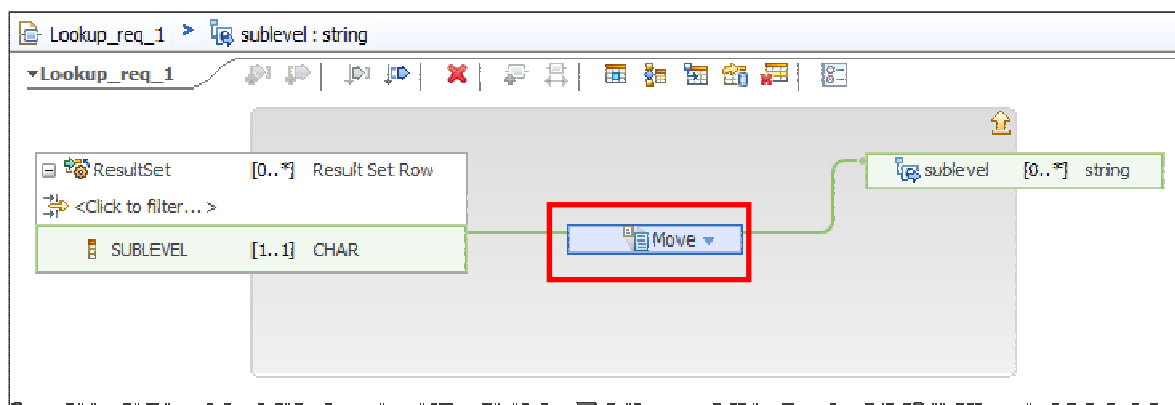


22. Click the “Select” link on the mapping you have just connected:



This will take you to a mapping editor for the “Select” result.

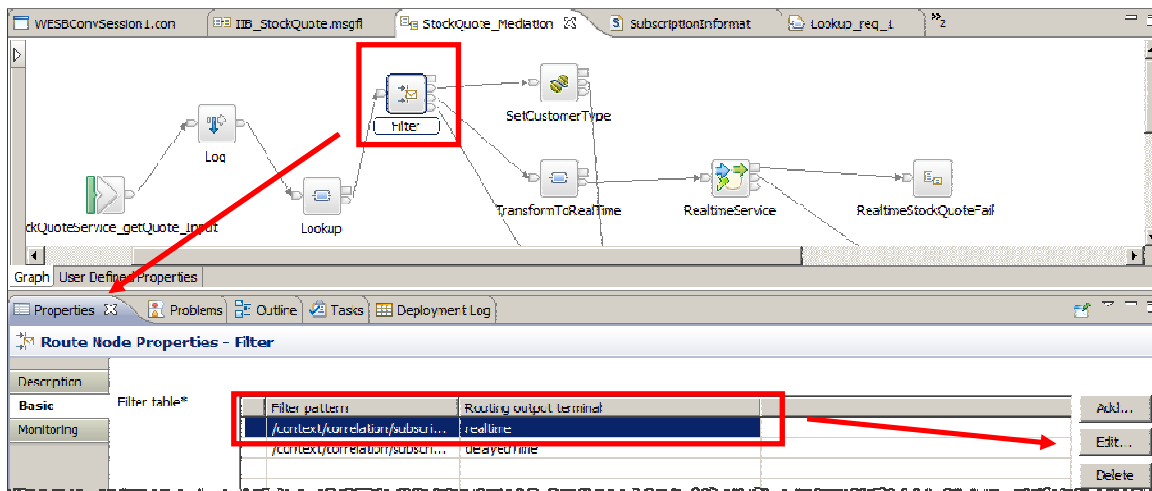
23. Map the SUBLEVEL input from the “ResultSet” input to the “sublevel” variable on the output. You should now have something like this:



Make sure the connection map is “Move”.

Save and close the mapping Node.

24. Click the node named “Filter”. This node is actually a Route Node, and is a decision point for the flow. The value “sublevel” mapped on the previous step will be compared and used to route the message to the correct node. Go to the properties tab of the “Filter” node:

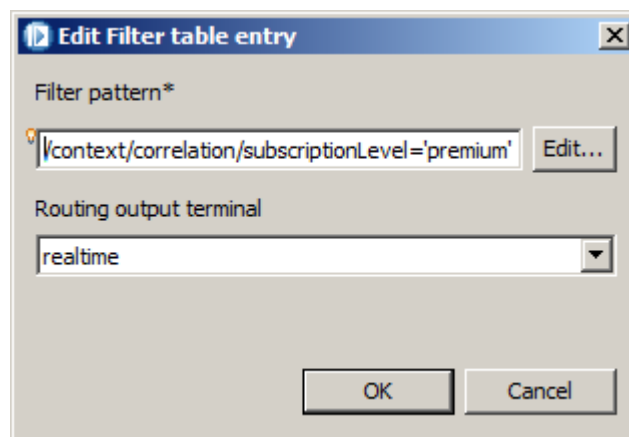


As the original StockQuote uses a different way to compare the Subscription Level, you have to change the filter to use the data from the defined message domains.

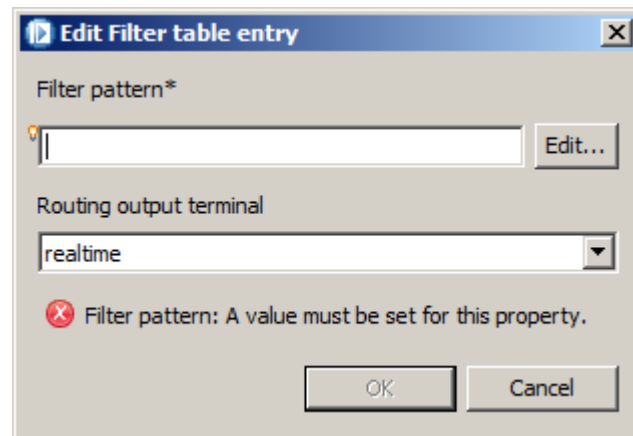
Select the first row of the “Filter Table” (the one routing to the “realtime” output terminal), and click the “Edit...” button.

In Windows Explorer, go to c:\student\WESB_Conversion\Resources and open the file **XPathExpressions.txt**

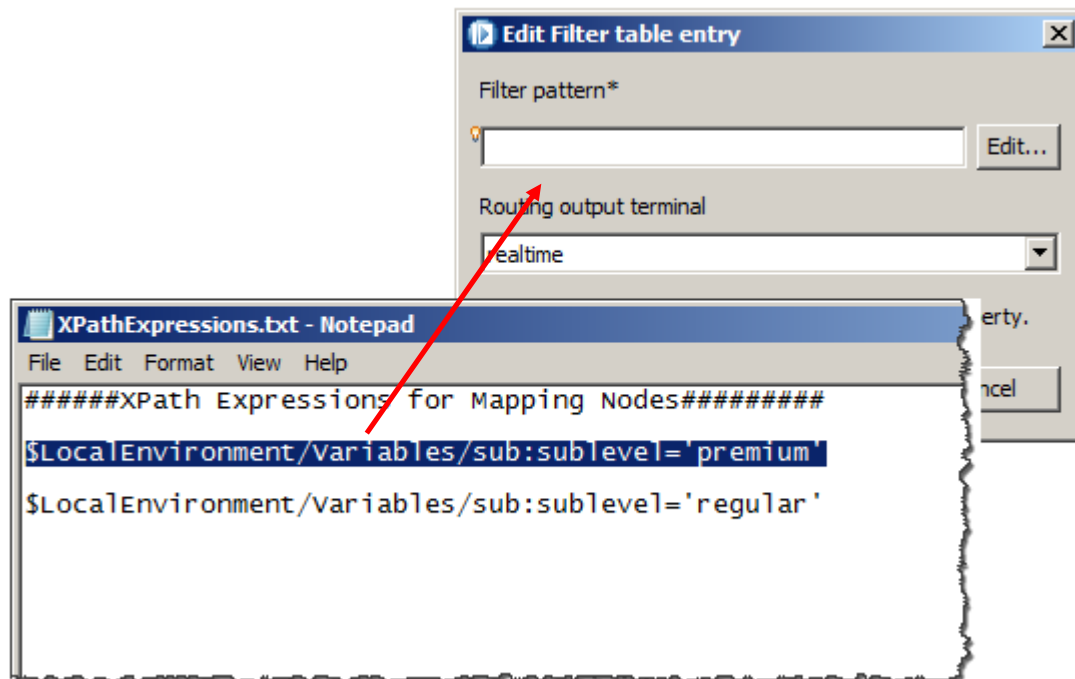
25. The “Edit Filter table entry” option will open:



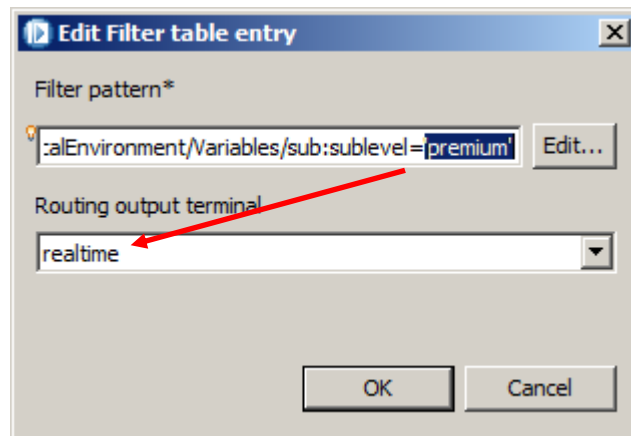
26. Clear the "Filter pattern"



27. Copy the value (" `$LocalEnvironment/Variables/sub:sublevel='premium'` ") from the text file and paste it on the "Filter pattern" box:



28. Make sure that the filter for '**premium**' is set to the "Routing output terminal" **realtime**:



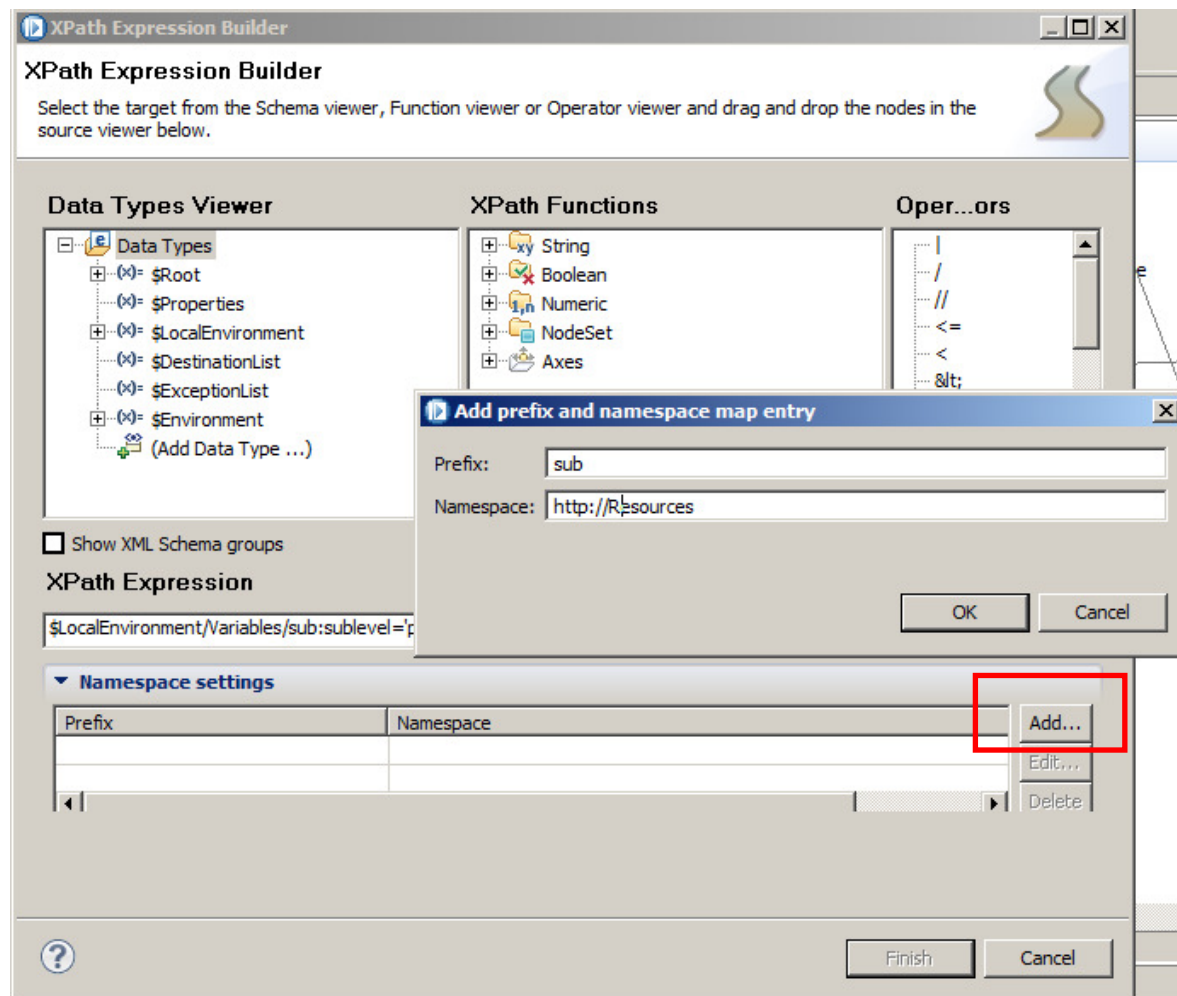
Do not click OK at this point.

29. The “sub” namespace reference by this XPath expression does not have an associated URI, so this needs to be created.

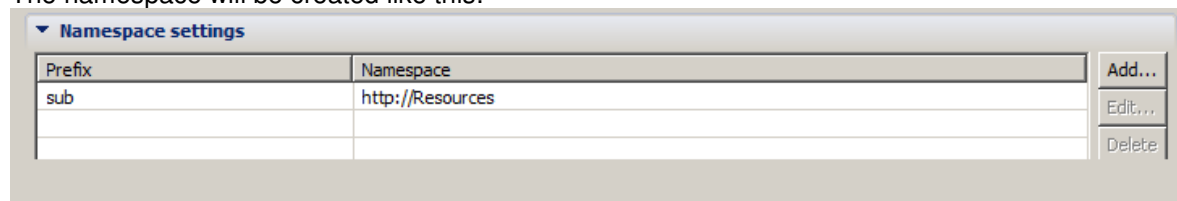
In the Edit Filter window (above) click the Edit button. This will open the XPath Expression builder.

Expand the Namespace settings, and click Add. In the dialogue window, set the Prefix to “sub”, and the namespace to <http://resources>.

Click OK, then Finish, then OK to return to the flow editor.



The namespace will be created like this:



30. Repeat the previous steps with the next row (delayedTime terminal). (Use the string you copied into the clipboard here. Change the 'premium' to 'regular').

You should have the filter table set like this:

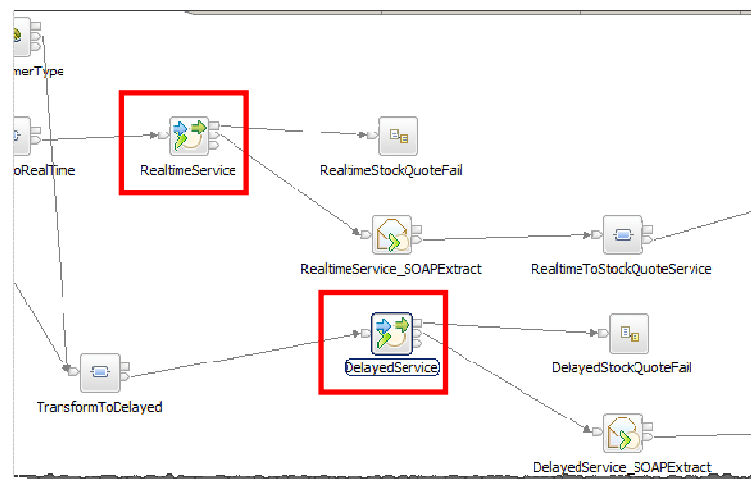
filter table*

Filter pattern	Routing output term
<code>\$LocalEnvironment/Variables/sub:sublevel='premium'</code>	realtime
<code>\$LocalEnvironment/Variables/sub:sublevel='regular'</code>	delayedTime

Save the filter. (Ctrl+s).

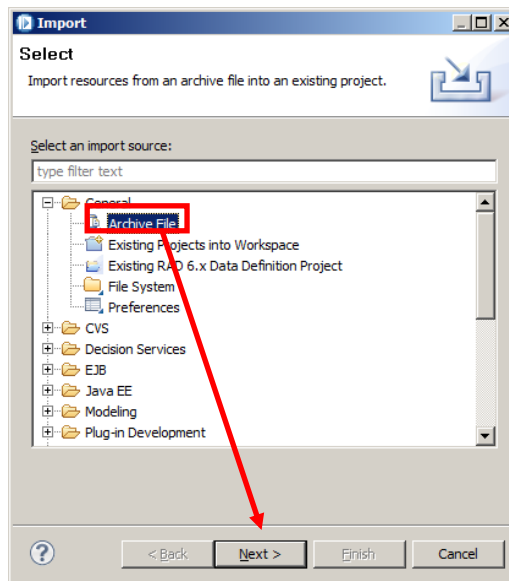
31. The original StockQuote application consumes the actual quote value from an application hosted on the WESB runtime. For this lab you are going to use an existing application running in WebSphere Application Server to obtain this quote value.

Find the **"RealTime"** and **"DelayedService"** SOAP Request Nodes (both converted from callouts to SOAP Request):

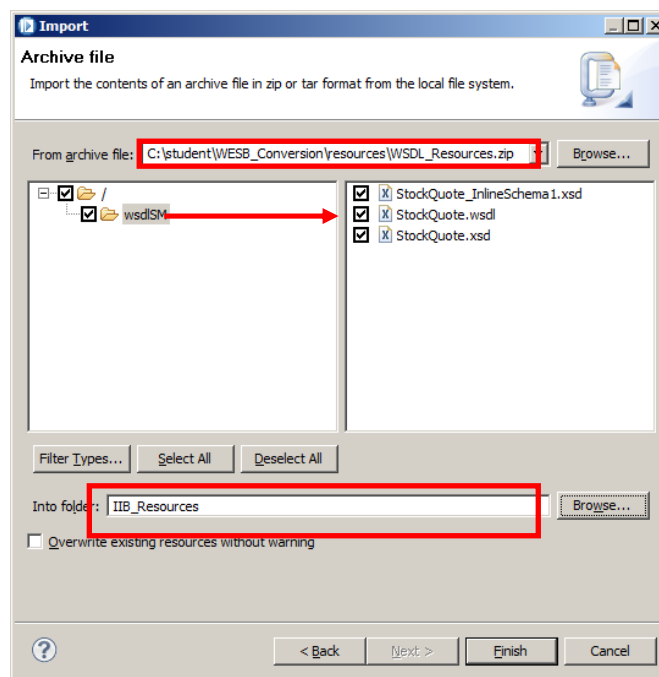


32. These nodes will invoke a StockQuote web service to get the actual value of the quote, so you need to import some WSDL and XSD definitions into the IIB_Resources library. Highlight the library, then click File > Import...

Under General, select "Archive File". Click "Next..."



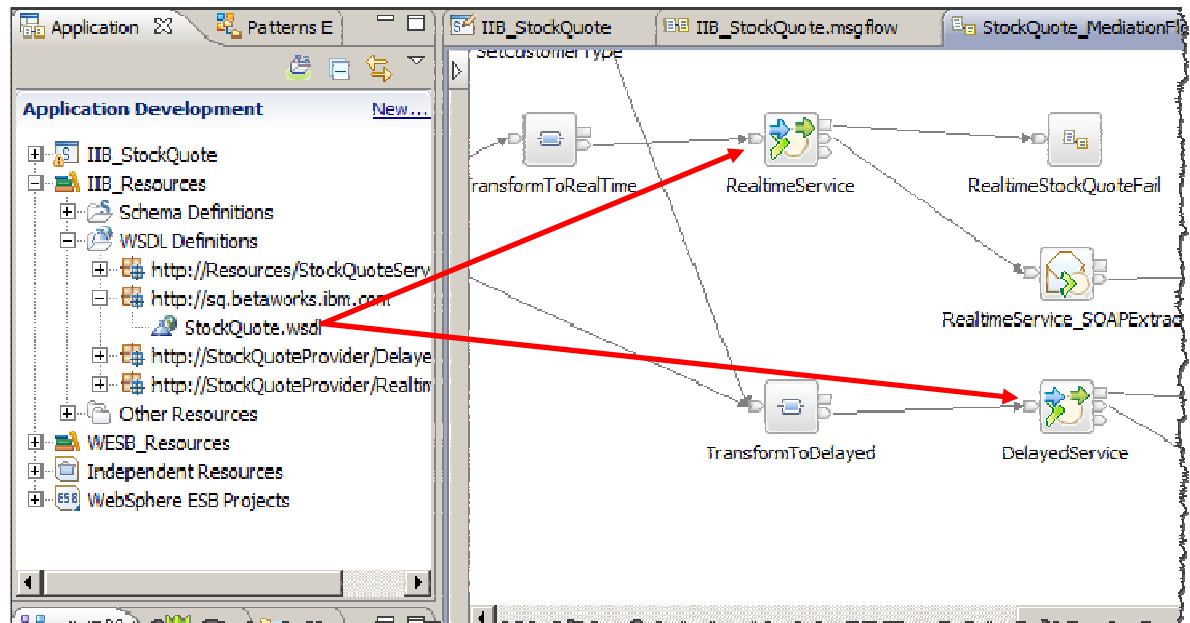
33. Browse for: C:\student\WESB_Conversion\resources\WSDL_Resources.zip. In the archive file selector, highlight wsdlSM and make sure all the three files are selected. Place the files into "IIB_Resources":



Click Finish.

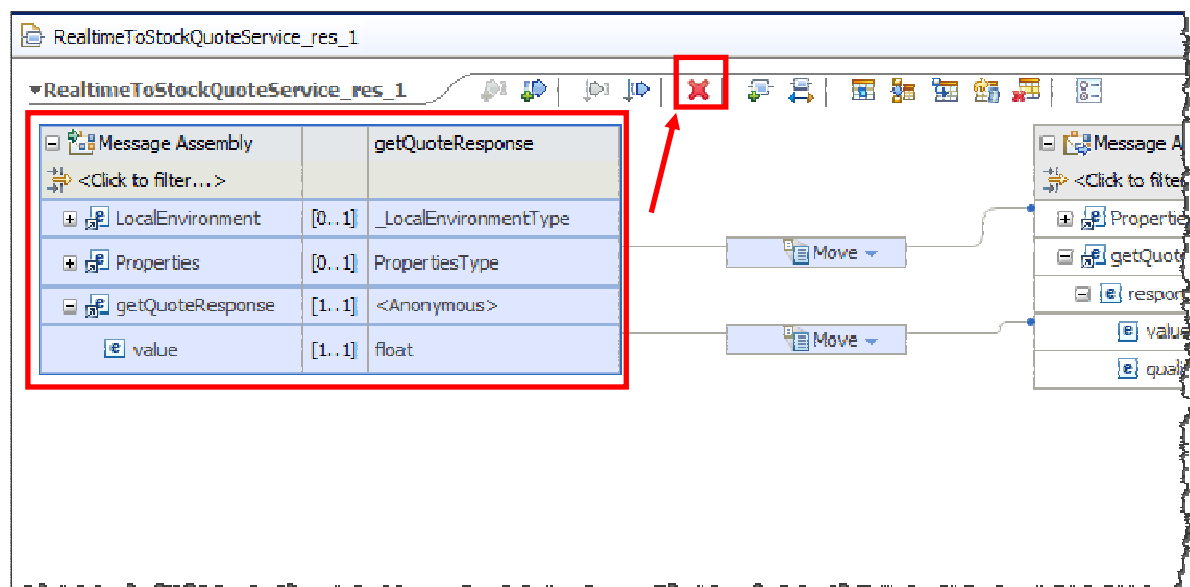
34. In the IIB_Resources library, expand WSDL Definitions and <http://sq.betaworks.ibm.com>.

Drag and drop the “**StockQuote.wSDL**” onto each SOAP Request Node.

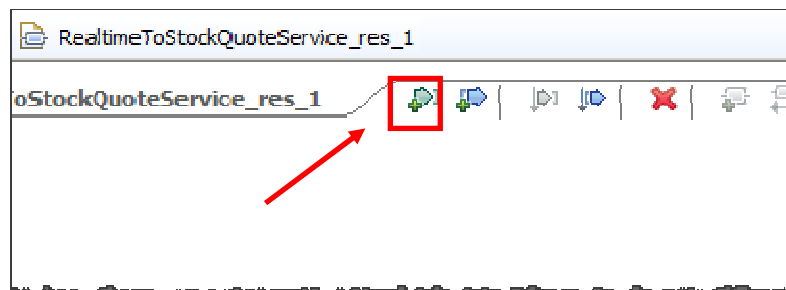


Save the updated flow.

35. Open the “**RealtimeToStockQuoteService**” mapping node. Delete the input Message Assembly. Select “Message Assembly” and click the “Delete selected elements” (red “X”) icon:

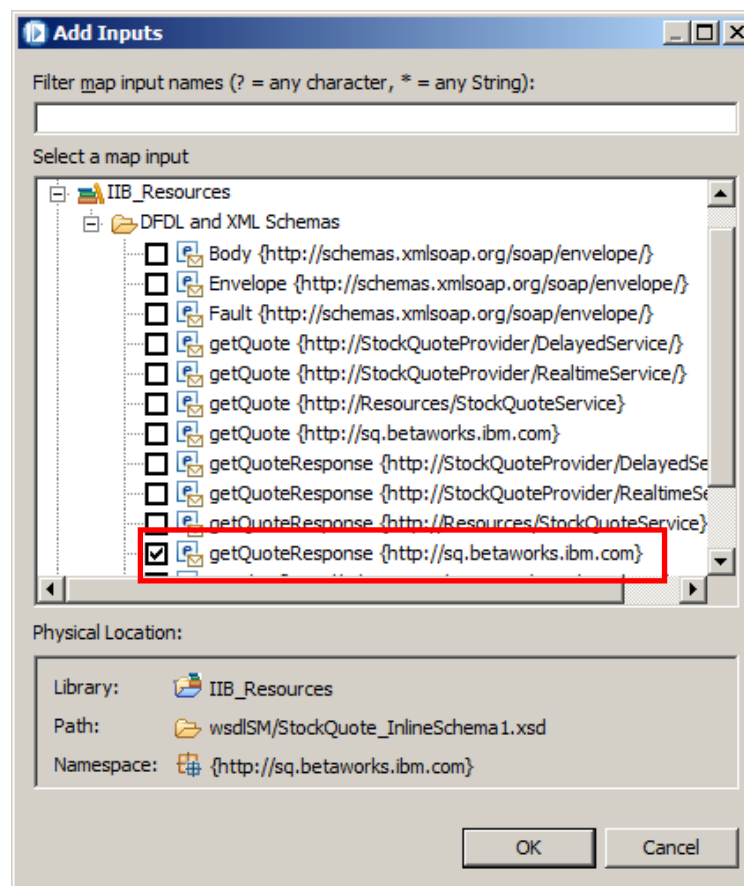


36. Click the “Add an input object” icon:



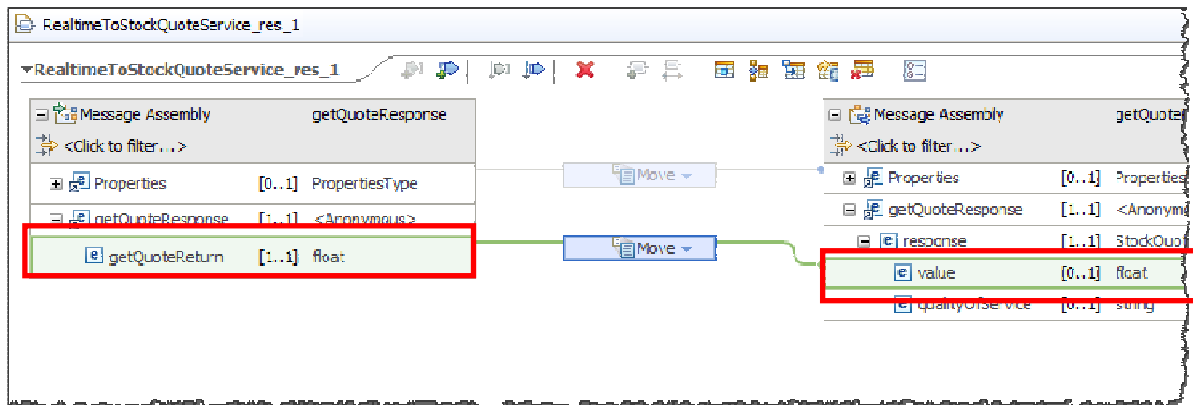
An “Add inputs” dialog will appear.

37. Expand IIB_Resources > “DFDL and XML Schemas” and select **getQuoteResponse {http://sq.betaworks.ibm.com}**.



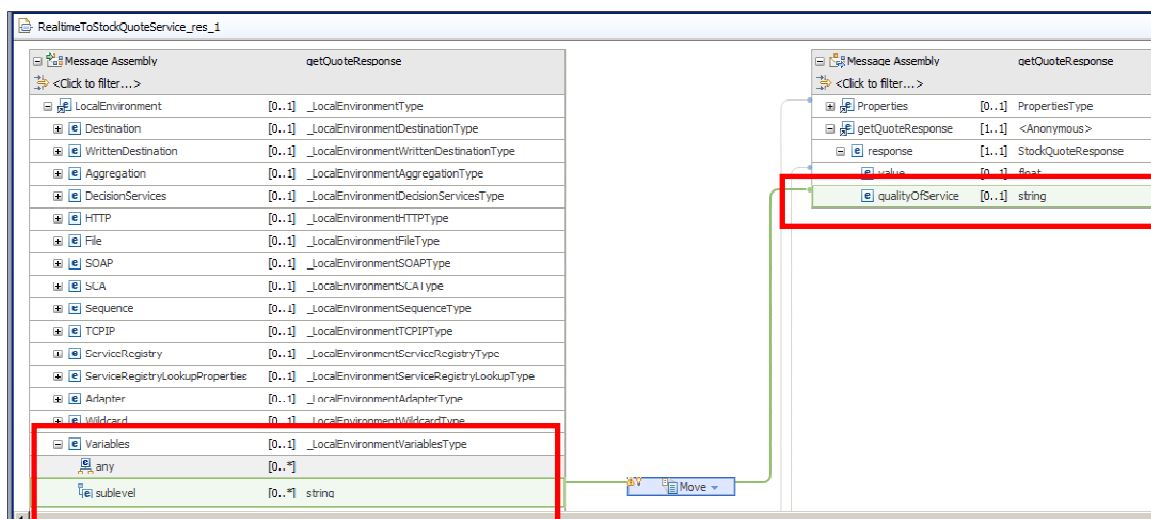
Click OK.

38. Expand the “getQuoteResponse” element and map **getQuoteReturn (float)** from the **input** Message Assembly to **value (float)** on the **output** Message Assembly:



39. Set the LocalEnvironment on the input message assembly, as you did earlier.

On the input message assembly, add the “sublevel” variable (in the same way that you added it to the output message before), and map the “**sublevel (string)**” variable from the **input** Message Assembly to “**qualityOfService (string)**” on the **output** Message Assembly.

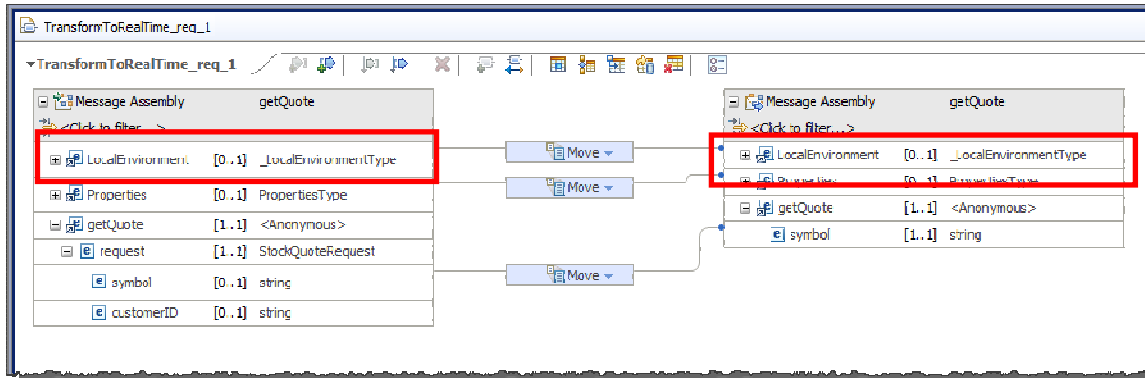


Make sure the mapping is set to “Move”.

Repeat the same steps (37. – 41.) for the “**DelayedToStockQuoteService**” node, and Save.

40. You're nearly there! You have to make some final changes to the remaining Mapping Nodes to allow the LocalEnvironment to be propagated correctly through the flow.

Open the “**TransformToRealTime**” Mapping Node. Add the LocalEnvironment to both **Input** and **Output** as before. This time map the LocalEnvironment entirely from the input to the output message assembly.

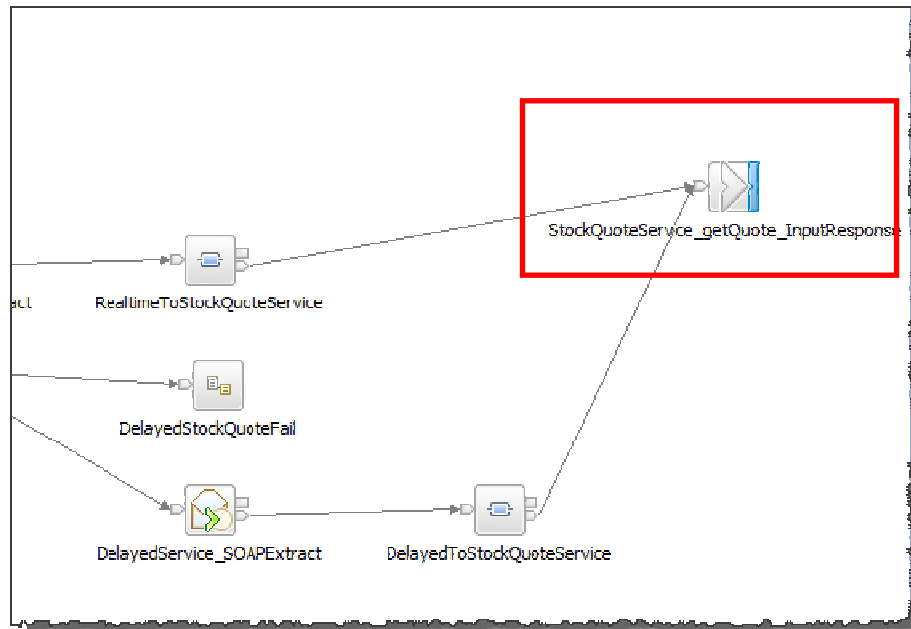


Make sure the mapping is set to “Move”.

41. Repeat the same steps for the “**TransformToDelayed**” node, and Save.

42. Finally, delete the “SetQualityOfService” JavaCompute Node. Connect the “RealTimeToStockQuoteService” and “DelayedToStockQuoteService” to the “StockQuoteService_getQuote_InputResponse” output node:

You can do this because the flow already has the Subscription Level variable for our customer on the local environment, and you don’t need to set this again in the flow.



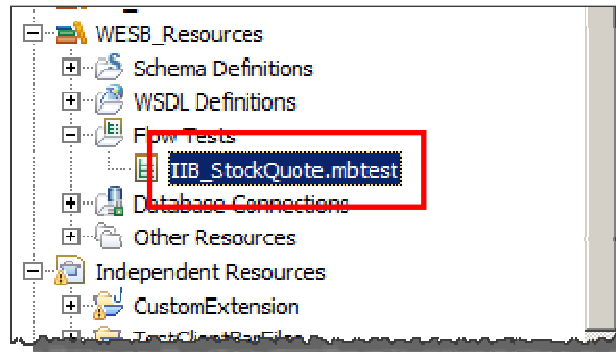
As we already have the Subscription Level variable for our customer on the local environment we don’t need to set this again in the flow.

One key feature of the Conversion procedure is that it allows you to keep the integration logic but use the IIB model to preserve the variables and message information as you are doing here with the Subscription Level information,.

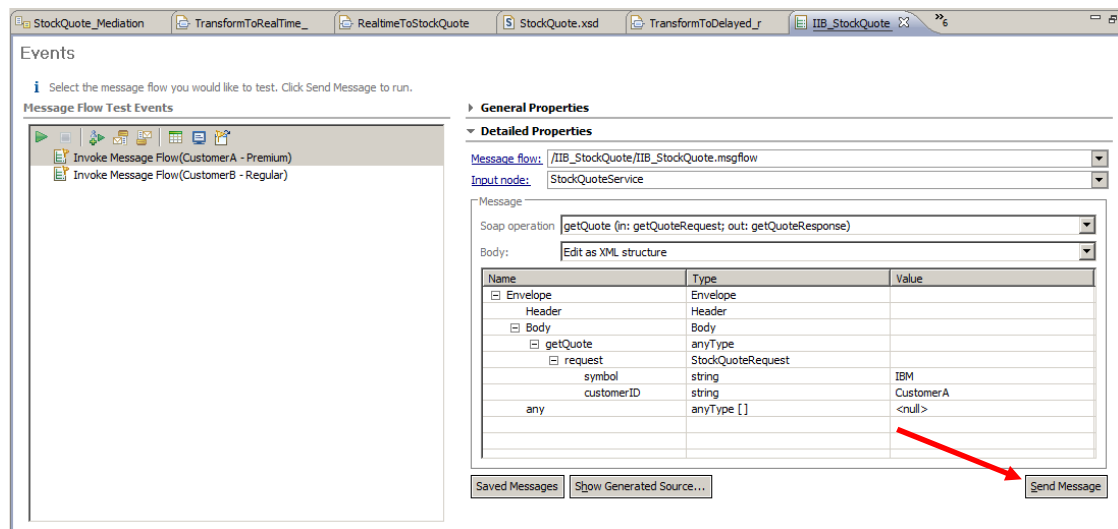
Save the flow one last time.

4. Test the IIB_StockQuote Service

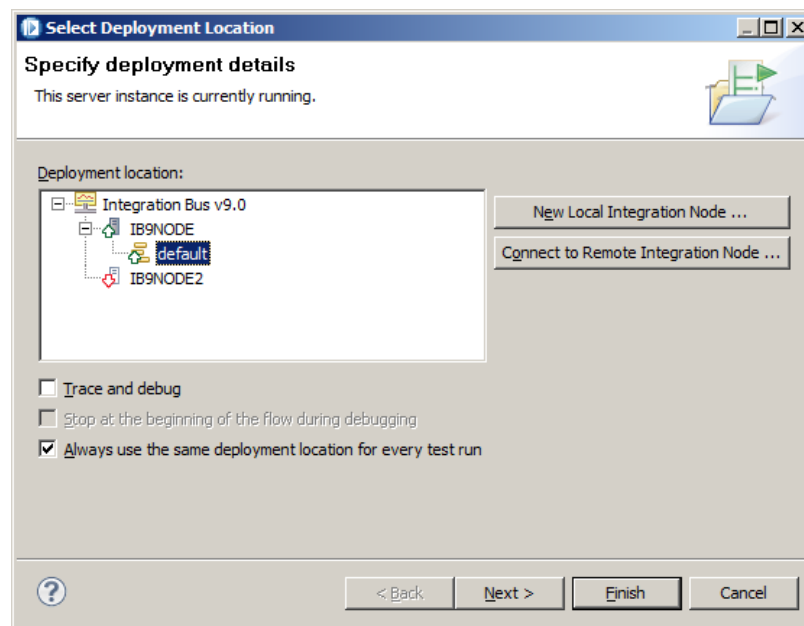
1. Expand the WESB_Resources folder and expand Flow Tests. Open IIB_StockQuote.mbttest.



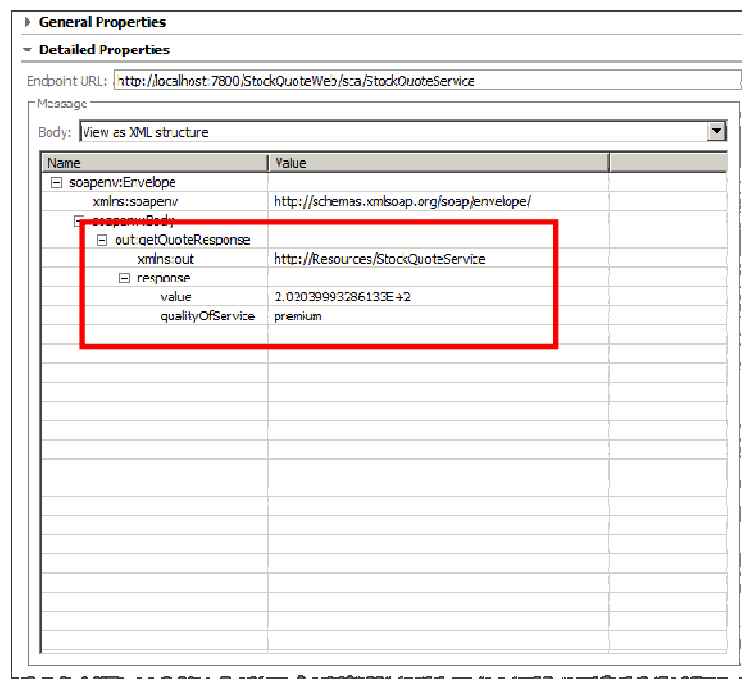
2. There are two test messages. One for a Premium customer (CustomerA) and another for a Regular customer (CustomerB). Click “Send Message”.



3. A dialog window will pop. Select the “default” Integration Server. Click “Finish” button.



4. You should receive this response for CustomerA:



This concludes the WESB conversion lab.

5. Appendix I

Mediation Primitive Converters

Mediation Primitive	Convert to	Usage	Converter class
Callout	Request (for example SOAPRequest)	StockQuote_MediationFlow.component	Built-in converter
CalloutResponse	Request (for example SOAPRequest)	StockQuote_MediationFlow.component	Built-in converter
ErrorInput	Subflow placeholder	StockQuote_MediationFlow.component	Placeholder converter
Fail	Subflow placeholder	StockQuote_MediationFlow.component	Placeholder converter
Input	Input	StockQuote_MediationFlow.component	Built-in converter
InputResponse	Reply (for example SOAPReply)	StockQuote_MediationFlow.component	Built-in converter
MessageElementSetter	JavaCompute	StockQuote_MediationFlow.component	Built-in converter
MessageFilter	Route	StockQuote_MediationFlow.component	Built-in converter
MessageLogger	Subflow placeholder	StockQuote_MediationFlow.component	Placeholder converter
XSL Transformation	Map	StockQuote_MediationFlow.component	Built-in converter

Export and Import Binding Converters

Binding	Convert to	Usage	Converter class
Jax/Ws Export	SOAPInput	StockQuoteService.export	Com.ibm.etools.mft.conversion.esb.extension.binding.JaxWsExportConverter
Jax/Ws Import	SOAPRequest	DelayedService.import, RealtimeService.import	Com.ibm.etools.mft.conversion.esb.extension.binding.JaxWsImportConverter